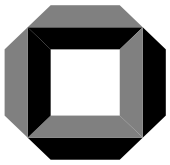

Universität Karlsruhe
Fakultät für Informatik
76128 Karlsruhe

Netzwerk-Management und Hochgeschwindigkeits-Kommunikation

Teil XIV

Seminar SS 1996

Herausgeber:
Stefan Dresler
Günter Schäfer
Hajo Wiltfang

Universität Karlsruhe
Institut für Telematik

Interner Bericht 23/96

Zusammenfassung

Der vorliegende Interne Bericht enthält die Beiträge zum Seminar „Netzwerk-Management und Hochgeschwindigkeits-Kommunikation“, das im Sommersemester 1996 zum 14. Mal stattgefunden hat.

Die Themenauswahl kann grob in folgende fünf Blöcke gegliedert werden:

1. Ein Block ist dem Management von Netzen gewidmet. Hier werden zum einen allgemeine Ansätze wie TMN und TINA vorgestellt und der Einsatz objektorientierter Techniken im Management diskutiert. Zum anderen finden sich Beiträge zum Management in ATM-Netzen. Themen sind hier das ILMI, Aspekte des Accounting sowie das Fehler- und Leistungsmanagement in ATM.

2. Ein zweiter Block beschäftigt sich mit Fragen im Zusammenhang von Multicast in ATM-Netzen. Hier werden der Mechanismus der Adreßauflösung mittels des MARS-Konzepts sowie die reihenfolgetreue Auslieferung in Multicast-Gruppen diskutiert.

3. Ein weiteres Thema ist ein Vergleich aktueller Authentisierungs- und Zertifizierungssysteme und liegt damit im Bereich der Sicherheit von Systemen.

4. Der vierte Themenbereich des Seminars beschäftigt sich mit dem ATM-Adaptionsschichtprotokoll SSCOP, das oft zur Unterstützung von Signalisierungsmechanismen wie denen des Q.2931 eingesetzt wird.

5. Schließlich werden noch Aspekte verteilter Simulationen betrachtet.

Abstract

This Technical Report includes student papers produced within small lessons called seminar of “Network Management and High Speed Communications”. For the fourteenth time this seminar has attracted a large number of diligent students, proving the broad interest in topics of network management and high speed communications.

The topics of this report may be divided into five blocks:

1. One block is devoted to the management of networks. Firstly, general approaches like TMN and TINA are introduced, and the deployment of object-oriented techniques in management is discussed. Secondly, there are contributions to the management of ATM networks. Topics are the ILMI, aspects of accounting, and the fault and performance management in ATM.

2. The second block deals with questions regarding multicast in ATM. Here, the MARS mechanism for the resolution of addresses and aspects of ordered multicast communication are discussed.

3. Another topic is a comparison of recent authentication and certification systems, taken from the area of security in communication systems.

4. The fourth area covered in the seminar deals with the ATM adaptation layer protocol SSCOP, which is often used to support signalling mechanisms like those specified in ITU Q.2931.

5. Finally, aspects of distributed simulations are dealt with.

Inhaltsverzeichnis

Vorwort

Das Seminar „Netzwerk-Management und Hochgeschwindigkeits-Kommunikation“ erfreute sich in den letzten Jahren immer größerer Beliebtheit. Gerade heutzutage sind Stichworte wie „ATM“, „Quality of Service“, „Mobil-Kommunikation“ oder „Internet“ in aller Munde. Daher sind die Forschungsgebiete in diesen Bereichen auch von allgemeinem Interesse, so daß sie eine derartige Vielzahl von innovativen Arbeiten aufweisen können, deren Behandlung in anderen Lehrveranstaltungen so detailliert nicht möglich ist.

Jetzt liegt auch der nunmehr vierzehnte Seminarband als Interner Bericht vor. Durch die engagierte Mitarbeit der beteiligten Studenten konnte so zumindest ein Ausschnitt aus dem komplexen und umfassenden Themengebiet klar und übersichtlich präsentiert werden. Für den Fleiß und das Engagement der Seminaristen sei daher an dieser Stelle recht herzlich gedankt.

Die ausgesprochen gute Resonanz bei den Studenten hat uns veranlaßt, auch im Wintersemester 1996/97 ein derartiges Seminar — natürlich mit geändertem aktuellem Inhalt — durchzuführen, so daß bald ein weiterer Interner Bericht mit neuen Forschungsergebnissen aus innovativen Tagungsbeiträgen erscheinen wird. Doch vorerst sollen im vorliegenden Band folgende Themengebiete vorgestellt werden:

Telecommunication Management Network

Der Betrieb und die Wartung moderner Telekommunikationsnetze kann nur durch umfassende technische Unterstützung ökonomisch realisiert werden. Diese Unterstützung soll zukünftig verstärkt durch Telekommunikations-Management-Netze (TMN) erbracht werden. An der Standardisierung solcher Managementnetze wird bereits seit 1986 auf internationaler Ebene gearbeitet. Der Beitrag gibt eine Einführung in die bisher erarbeiteten Konzepte von TMNs und einen Überblick über die wichtigsten Standardisierunggremien in diesem Bereich.

Vergleich aktueller Authentisierungs- und Zertifizierungssysteme

Mit der zunehmenden Vernetzung von sowohl Rechnern zu lokalen Netzen als auch der Verknüpfung der lokalen Netze zu einem globalen Internetzverbund über Weitverkehrsnetze steigt der Bedarf an Sicherheitsdiensten, die sicherstellen, daß Rechnerressourcen nur von den Personen genutzt werden können, die dazu berechtigt sind. Authentisierung ist der Sicherheitsdienst, mit dem die Identität von Instanzen überprüft werden kann. Dieser Beitrag stellt aktuelle Authentisierungs- und Zertifizierungssysteme gegenüber und vergleicht ihre spezifischen Stärken und Schwächen miteinander.

Einsatz objektorientierter Techniken im Netzwerk- und Anwendungsmanagement

Der Beitrag gibt eine Einführung in objektorientierte Konzepte und beschreibt ihre Nutzung in den Bereichen des Netzwerk- und Anwendungsmanagements.

Das Adaptionsschichtprotokoll SSCOP

Der von der ATM-Schicht zur Verfügung gestellte Dienst ist nicht zuverlässig; es können bei der Übertragung beispielsweise Zellen verlorengehen. Das Protokoll SSCOP (Service Specific Connection Oriented Protocol) kann als SSCS (Service Specific Convergence Sublayer) der ATM-Adaptionsschicht eingesetzt werden und bietet an seiner Dienstschnittstelle einen gesicherten Dienst. Dieser wird etwa von Signalisierungsprotokollen genutzt.
Dieser Beitrag stellt die Funktionsweise von SSCOP vor und geht auch auf dessen Leistungsfähigkeit ein.

Adreßauflösung für Multicastgruppen mit MARS

Das Internet Protocol (IP) stellt einen verbindungslosen Multicast-Dienst zur Verfügung, der eine 1:N-Kommunikation im Internet ermöglicht. Mit dem MARS (Multicast Address Resolution Server) liegt ein Mechanismus vor, der diesen Dienst auf den verbindungsorientierten Dienst, der vom User Network Interface (UNI) 3.0/3.1 von ATM angeboten wird, abbildet.
Die verwendeten Mechanismen sowie ein typischer Ablauf einer Mehrpunkt-Kommunikation werden in diesem Beitrag vorgestellt. Darin wird auch auf die zwei möglichen Realisierungsformen für Multicast über IP mittels MARS — ein Netz von Multicast ATM-Verbindungen bzw. Nutzung sogenannter Multicast Server (MCS) — eingegangen.

Reihenfolgetreue Auslieferung in Multicast-Gruppen

Arbeiten beispielsweise mehrere Benutzer mit einer Datenbank, die auf einem zweiten Rechner gespiegelt wird oder die sogar verteilt realisiert ist, so ist es unbedingt erforderlich, daß Änderungsaufträge bei alle Teilen der Datenbank in der gleichen Reihenfolge durchgeführt werden. Dabei kann ein Dienst hilfreich sein, der eine gesicherte, reihenfolgetreue Auslieferung von Nachrichten im Multicast- (1:N-) oder Multipeer- (M:N-) Fall sicherstellt.
Dieser Beitrag stellt verschiedene Vorgehensweise vor, mit denen ein solcher Dienst bereitgestellt werden kann. Dabei wird auch auf Fragen der Leistungsanalyse eingegangen.

Aspekte verteilter Simulationen

Ereignisgesteuerte Simulationen bieten für bestimmte Fragestellungen eine gute Hilfe zur Analyse von Systemen, die nicht und nur sehr schwierig analytisch zu erfassen sind. Solche Simulationen können mitunter sehr umfangreich werden. Aus diesem Grund ist man bestrebt, solche Simulationen auf mehrere Rechner zu verteilen.
Der Beitrag beginnt mit einer Vorstellung von Grundlagen ereignisgesteuerter Simulationen. Anschließend wird auf Aspekte der Verteilung, insbesondere mögliche Realisierungsformen, und die damit verbundenen Schwierigkeiten eingegangen.

Interim Local Management Interface

Vom ATM Forum wurde für das Management von privaten ATM-Netzwerken ein Management-Interface festgelegt. Dieses Interim Local Management Interface (ILMI) basiert auf dem Protokoll SNMP, dem Managementprotokoll des Internets. Dieser Beitrag stellt die ILMI-Schnittstelle vor. Dabei wird zum einen die Schnittstelle selbst sowie das Protokoll behandelt, zum anderen wird die ILMI-MIB, die Management Information Base mit allen Managementobjekten der ILMI-Schnittstelle, vorgestellt. Hierbei ist vor allem der abschließende Vergleich mit der ATM-MIB der IETF von Interesse.

Fehler- und Leistungsmanagement in ATM-Netzen

Zwei Bereiche des Netzwerkmanagements, das Fehler- und das Leistungsmanagement, werden in dem vorliegenden Beitrag in Bezug auf ATM-Netzwerke untersucht. Grundlegende Funktionen zu diesen beiden Bereichen des Managements von ATM-Netzwerken werden mit Hilfe von OAM-Zellen (Operation, Administration and Maintenance) realisiert, die parallel zu den eigentlichen Nutzdaten auf dem ATM-Netzwerk übertragen werden können. Der Beitrag beinhaltet einerseits eine Vorstellung aller Funktionen, die durch OAM-Zellen momentan zur Verfügung gestellt werden. Andererseits wird der Vorschlag für eine MIB (Management Information Base) behandelt. Diese MIB definiert Objekte für das Testing von ATM-Netzen. Dabei werden für die Durchführung dieser Tests hauptsächlich OAM-Zellen benutzt.

Accounting in ATM-Netzen

Bezogen auf das Netzwerkmanagement läßt sich Accounting in den Teilbereich des Abrechnungsmanagements einordnen. Die wesentliche Aufgabe dabei ist, Daten über die Benutzung des Netzwerks und seiner Ressourcen zu erfassen. Diese Daten werden von den ATM-Systemen gesammelt und in einer MIB dem Netzwerkmanagement zur Verfügung gestellt. Neben der Beschreibung des Accountings in ATM-Netzen beschäftigt sich dieser Beitrag im wesentlichen mit einem Vorschlag für eine Accounting-MIB.

Telecommunications Information Networking Architecture (TINA)

Die Telecommunications Information Networking Architecture wurde 1990 im Rahmen eines eigens dafür durchgeführten Workshops entwickelt. Ziel des gegründeten TINA-Konsortiums ist es, eine allgemeine Architektur festzulegen, welche die Bereiche der Informatik und der Telekommunikation zusammenbringt. Dieser Beitrag beschreibt zum einen die bis heute festgelegte TINA-Architektur und stellt zum anderen das TINA Network Resource Model vor.

Telecommunication Management Network

Thomas Dietz

Kurzfassung

Dieser Artikel gibt einen Überblick über die Architektur eines Telekommunikations-Management-Netzwerkes. Er stellt das funktionale Architektur-Modell vor, das die Funktionalität des Management Netzwerkes in verschiedene Blöcke aufteilt. Die physikalische Architektur integriert die verschiedenen Funktionsblöcke in Geräten und definiert die Schnittstellen zwischen den Geräten. Desweiteren werden einige ausgewählte Funktionseinheiten und Funktionen beschrieben, die Informationen zwischen verschiedenen Netzwerkelementen transportieren. Außerdem wird noch kurz auf die TMN Schichten und die Verteilung von TMN-Funktionen eingegangen. Am Schluß werden noch einige Gremien und Organisationen vorgestellt, die an der Standardisierung der Telekommunikations-Management-Netzwerke arbeiten.

1 Einleitung

Telekommunikationsnetzwerke werden jeden Tag größer und „intelligenter“ und ihre Verteilung nimmt immer mehr zu. Die Netzwerke entwickeln sich zu softwarebasierten Systemen mit tausenden von intelligenten Netzwerkelementen, die eine Vielzahl von Diensten bereitstellen. Um diese Netzwerke zu betreiben und zu verwalten ist eine entsprechende Infrastruktur nötig. Eine solche Infrastruktur soll durch ein Managementnetzwerk mit Standard-Protokollen, Schnittstellen und Architekturen geschaffen werden, das von der International Telecommunications Union–Telecommunications (ITU-T – früher CCITT) als *Telecommunications Management Network* (TMN) bezeichnet wird.

Das TMN soll die Funktionen für den Betrieb, die Verwaltung und die Instandhaltung des Telekommunikationsnetzwerks bereitstellen. Diese Funktionen können in zwei Gruppen eingeteilt werden – die Basisfunktionen und die erweiterten Funktionen. Die Basisfunktionen sind

- Managementfunktionen wie Konfigurationsmanagement, Leistungsmanagement, Fehlerbehandlung, Accounting-Management und Sicherungsfunktionen;
- Kommunikationsfunktionen zwischen den verschiedenen Funktionseinheiten;
- und Plannungsfunktionen, unter anderem auch die physikalische Plannung des Netzes und Arbeitsplannung.

Diese Basisfunktionen dienen als Grundlage für die erweiterten Funktionen, die dann Funktionen wie Dienstmanagement, Wiederherstellung des Netzwerkes, Rekonfiguration und Bandbreitenzuteilung ermöglichen.

Ich möchte in diesem Artikel TMN Modelle, Architekturen, Standardschnittstellen, Kommunikationsfunktionen und auch die Verteilung der TMN-Funktionen vorstellen. Der Artikel kann aber nur einen Überblick geben, da ein TMN sehr verschiedene Ausprägungen haben kann.

2 Funktionales Architektur-Modell

Funktional gesehen ist das TMN ein eigenständiges Netzwerk, das die Verwaltung eines Telekommunikationsnetzwerkes ermöglicht. In der Realität wird es aber Teile des Telekommunikationsnetzwerkes mitbenutzen, um seine Funktionalität zu erbringen. Mit anderen Worten: Teile des TMN können ein im Telekommunikationsnetzwerk eingebettetes logisches Netzwerk sein.

Das TMN stellt nun die Mittel bereit, um die Informationen, die mit der Verwaltung eines Telekommunikationsnetzwerkes und seiner Dienste zusamenhängen, zu transportieren und zu verarbeiten. Die Funktionalität kann im wesentlichen in drei Blöcke eingeteilt werden, den Betriebssystem Funktionsblock (*operations systems function* OSF), den Vermittlungsfunktionsblock (*mediation function* MF) und den Kommunikations Funktionsblock (*data communication function* DCF). Außerdem beinhaltet das TMN Teile der Netzwerkelement Funktionsblöcke (*network element function* NEF), der Arbeitsstationen Funktionsblöcke (*workstation function* WSF) und der Q Adapter Funktionsblöcke (*Q adaptor function* QAF) des Telekommunikationsnetzwerkes. Die einzelnen Funktionsblöcke haben die folgende Funktionalität:

Operations Systems Functions Blocks: Diese Blöcke beinhalten die meisten der Basis- und der erweiterten Funktionen aus Abschnitt 1. Es werden viele verschiedene Blöcke benötigt um die ganzen Management- und Planungsfunktionen der heutigen Telekommunikationsnetzwerke zu erfüllen. Die ITU-T definiert vier OSF Blöcke, um verschiedene Schichten zu unterstützen, das Business Management Layer, das Service Management Layer, das Network Management Layer und das Network Element Management Layer auch Subnetwork Management Layer genannt (vgl. dazu Abschnitt 6.1).

Mediation Function Blocks: Die MF Blöcke erlauben den verschiedenen TMN-Funktionsblöcken, miteinander über verschiedene Schnittstellen bzw. Referenzpunkte zu kommunizieren. Die Grenzen zwischen OSF und MF Blöcken sind nicht klar gezogen, da die ITU-T auch Verarbeitungs- und Transportfunktionen für die MF Blöcke definiert. Beispiele für MFs sind Informationstransportfunktionen wie Protokoll-, Signalkonvertierung, und Routing und Informationsverarbeitungsfunktionen wie Ausführung, Speicherung und Filterung.

Data Communications Function Blocks: Die Hauptfunktionalität dieser Blöcke ist der Informationstransport für die Kommunikation zwischen den verschiedenen TMN-Blöcken.

Network Element-Network Management Function Blocks: Da die Netzwerkelemente in den letzten Jahrzehnten immer intelligenter geworden sind, wurden viele OSF und MF Funktionen in die NEs integriert. Beispiele für NE-NM Funktionen sind Protokollkonvertierung, Routing, Fehlerlokalisierung und Datenbackup.

Workstation Network Management Function Blocks: Die WS-NMF Blöcke stellen die Funktionen für die Benutzerinteraktion zur Verfügung.

Q Adaptor Network Management Function Blocks: Die Blöcke konvertieren zwischen TMN Referenzpunkten und Nicht-TMN Referenzpunkten.

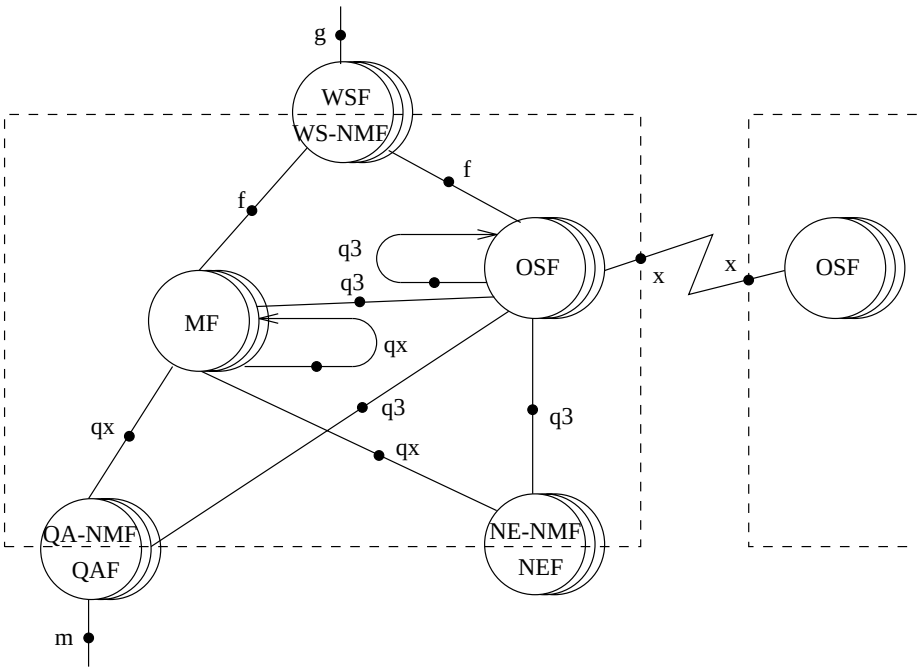

Abbildung 1. TMN Funktionsblöcke und Referenzpunkte.

Die Verbindungen zwischen den einzelnen Funktionsblöcken sind Referenzpunkte. Ein Referenzpunkt ist ein konzeptioneller Punkt an dem Information zwischen sich nicht überlappenden Blöcken ausgetauscht wird. Der Referenzpunkt wird zur Schnittstelle, wenn die verbundenen Funktionsblöcke in getrennten physikalischen Geräten untergebracht sind. Abbildung 1 zeigt die Funktionsblöcke im TMN mit ihren Referenzpunkten. Im TMN sind die folgenden Referenzpunkte definiert:

q Referenzpunkt: Er verbindet die TMN OSF, MF, NEF, und QAF Funktionsblöcke entweder direkt oder über DCF. Dabei verbindet der q3 Referenzpunkt NEF, MF, QAF und OSF mit OSFs und der qx Referenzpunkt MF mit MF, NEF und QAF.

f Referenzpunkt: Er verbindet OSF und MF Funktionsblöcke mit dem WSF Funktionsblock.

x Referenzpunkt: Er verbindet OSF Funktionsblöcke in verschiedenen TMNs oder auch einen OSF Funktionsblock mit einer gleichwertigen Einrichtung in einer Nicht-TMN Umgebung.

g Referenzpunkt: Er gehört eigentlich nicht mehr zum TMN, obwohl er Informationen des TMNs überträgt. Der g Referenz Punkt ist außerhalb des TMNs zwischen WSF und dem Benutzer anzusiedeln.

m Referenzpunkt: Er ermöglicht es Netzwerkelemente zu managen, die nicht mit dem TMN konform sind. Er ist ebenfalls außerhalb des TMNs anzusiedeln und stellt die Verbindung zu den nicht TMN-konformen Einheiten dar.

3 Physikalische Architektur

Die TMN-Funktionen können in vielfältiger Weise in einer physikalischen Architektur implementiert werden. Die physikalische Architektur schafft die Grundlage, um die Informationen für das Management des Telekommunikationsnetzwerkes zu transportieren und zu verarbeiten. Eine solche Architektur besteht aus Betriebssystemen (*Operations Systems* OSs), Kommunikationsnetzwerken (*Data Communications Networks* DCNs), Vermittlungsgeräten (*Mediation Devices* MDs), Arbeitsstationen (*Workstations* WSs), Netzwerkelementen (*Network Elements* NEs) und Q Adaptern (*Q Adaptors* QAs).

Da die Telekommunikationseinrichtungen u.a. durch den Einsatz von Mikroprozessoren immer komplexer werden, können die Architekturen sehr unterschiedlich ausfallen. Ein Beispiel für eine solche Architektur ist in Abbildung 2 gegeben. Es ist möglich, daß Teile wie z.B. MDs oder auch QAs fehlen. Auch die Ausprägungen des DCN reichen von einer einfachen Punkt-zu-Punkt-Verbindung bis zu einem komplexen paketvermittelnden Netzwerk.

Die Referenzpunkte aus dem funktionalen Architektur-Modell werden hier zu Schnittstellen, wenn die Funktionsblöcke in unterschiedlichen Geräten untergebracht sind. Die Verbindungen zwischen den einzelnen Einheiten (NEs, OSs, WSs, usw.) über ein DCN werden mit standardisierten Schnittstellen realisiert. Damit das TMN seine Managementfunktionen auch unabhängig von der Art der einzelnen Geräte und deren Hersteller erfüllen kann, müssen kompatible Kommunikationsprotokolle mit kompatiblen generischen Nachrichtendefinitionen vorhanden sein. Die interoperable Schnittstelle vereinfacht die Komunikationsprobleme in einem Netzwerk, das aus vielen Komponenten von verschiedenen Herstellern besteht. Sie definiert den Protokollturm und die Nachrichten, die mit diesem Protokoll verschickt werden. Sie basiert auf der objektorientierten Sichtweise der Kommunikation, und alle Nachrichten manipulieren Objekte. Die interoperable Schnittstelle ist also eine formal definierte Menge von Protokollen, Prozeduren, Nachrichtenformaten und Semantiken, die für die Managementkommunikation benutzt werden.

Die TMN Schnittstellen basieren auf den Konzepten des OSI-Referenzmodells. Die Protokolltürme sollten interoperable Schnittstellen unterstützen und sich, soweit es ökonomisch möglich ist, an das OSI-Schichtenmodell halten.

Die Qx Schnittstelle: Sie unterstützt nur eine kleine Anzahl von Funktionen und benötigt deshalb auch im allgemeinen nur die Schichten 1 und 2 des OSI-Referenzmodells. Sie wird häufig für einfache Funktionen wie das Ändern des Alarmstatus oder Rücksetzen von Alarmmeldungen usw. eingesetzt. Soll eine größere Anzahl von Funktionen unterstützt werden, so werden zusätzliche Protokolldienste von Schicht 3 bis hin zu Schicht 7 benötigt.

Die Q3 Schnittstelle: Sie unterstützt die komplexen Funktionen und benötigt deshalb meist die Protokolldienste von Schicht 1 bis 7. Manche Dienste können aus ökonomischen oder auch leistungsbedingten Gründen entfallen.

Die X Schnittstelle: Sie unterstützt die Funktionen zwischen einem OS aus einem

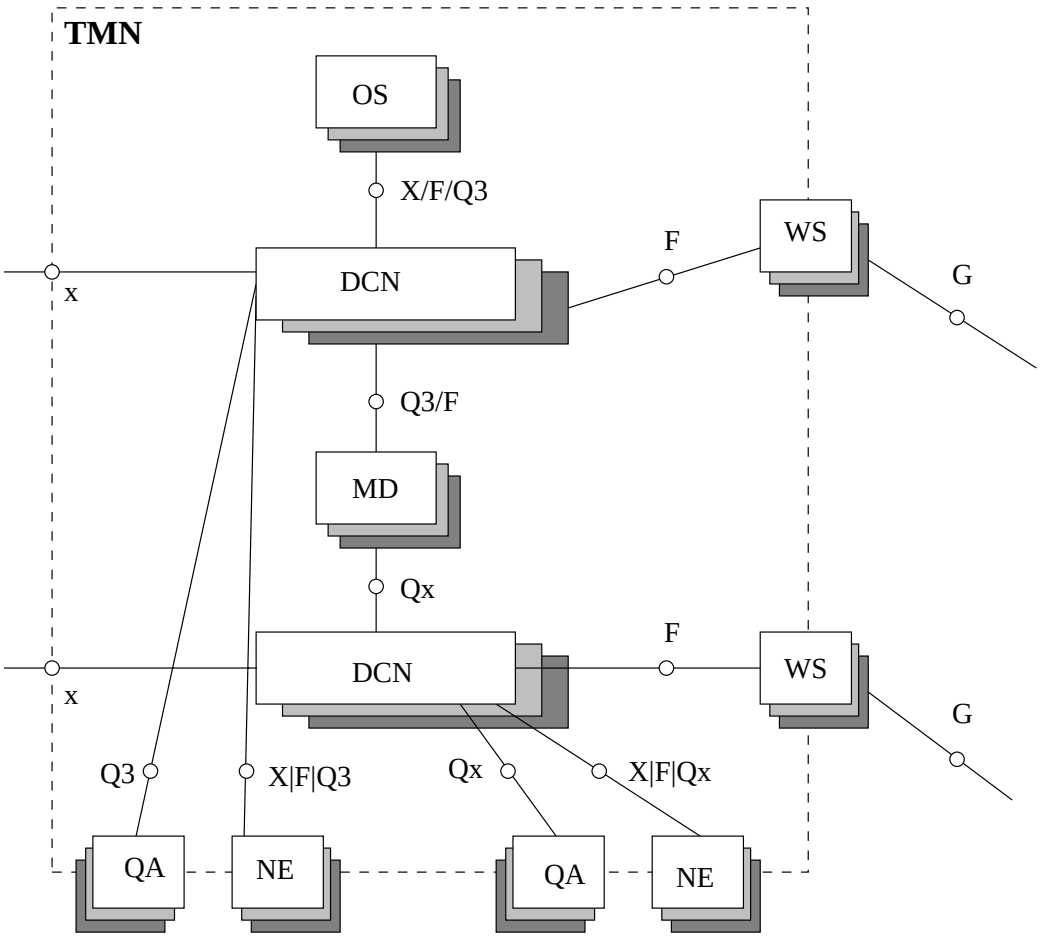

Abbildung 2. Eine einfache physikalische TMN Architektur.

TMN und einem OS aus einem anderen Management System. Sie benötigt ebenfalls die Protokolldienste der Schichten 1 bis 7, um ihre Aufgabe zu erfüllen. Sie kann u.U. auch als Ersatz für die Q3 Schnittstelle zwischen OSs eingesetzt werden.

Die F Schnittstelle: Sie unterstützt diejenigen Funktionen, die Arbeitsstationen über ein DCN mit den Komponenten verbindet, die OSF und MF Funktionalität enthalten.

4 Funktionseinheiten

In diesem Abschitt möchte ich einen kurzen Überblick über die wichtigsten TMN-Funktionseinheiten geben.

Embedded Communications Channels: Die bestehenden Standarddigitalsignale setzen sich aus Kanälen, die Nutzdaten transportieren, und aus Kanälen, die keine Nutzdaten transportieren, zusammen. Die eingebetteten Kommunikationskanäle (*Embedded Communications Channels* ECC) sind die Kanäle, die keine Nutzdaten transportieren. Das Wort „eingebettet“ bedeutet, daß die ECCs von Digitalsignalen im selben physikalischen Medium wie die Nutzdatenkanäle tranportiert werden. Hier sind die ECCs von Bedeutung, die Netzwerkmanagement-Anwendungen unterstützen. Die ECCs benutzen entweder schon existierende ungenutzte Bits in einem Digitalsignal

oder reservieren sich eigene Bits im Digitalsignal. Die Neustrukturierung bestehender Signalformate ermöglicht oft freigewordene Bitfelder für die ECCs zu nutzen. Manch neue Signalformate haben reservierte, ungenutzte Kapazitäten die für ECCs genutzt werden können. Wenn es notwendig ist, können auch ganze Kanäle in einem Digitalsignal als ECC genutzt werden.

Embedded Operations Channels: Die Teile der ECCs, die für TMN Funktionen und Anwendungen genutzt werden, sind die eingebetteten Operationskanäle (*Embedded Operations Channels* EOC). Man unterscheidet zwischen nachrichtenorientierten EOCs (message-oriented EOCs) und bitorientierten EOCs (bit-oriented EOCs). Bei nachrichtenorientierten EOCs werden die Informationen im Datenteil der höchsten Protokollebene transportiert. Die Nachrichten können hier sehr viele Bits umfassen. Im Gegensatz dazu sind die bitorientierten EOCs nur für einfache Nachrichten wie Fehler- oder Alarmanzeigen geeignet. Ihr Vorteil liegt darin, daß sie keine höheren Protokollschichten benötigen.

Operations Channel: Wenn für die Kommunikation zwischen Netzwerkelementen (NE zu NE bzw. NE zu OS) ein eigenes Übertragungsmedium zur Verfügung steht, nennt man dies einen Operationskanal (*Operations Channel* OC). Die Unterscheidung zwischen EOC und OC ist notwendig, da sie verschiedene physikalische Schnittstellen besitzen.

Shared Channel: Bei den *Shared Channels* (SC) teilen sich Dienste und Management Operationen einen Kanal im statistischen Mutiplexverfahren.

Network Element: Ein Netzwerkelement (*Network Element* NE) ist im allgemeinen ein prozessorgesteuerter Netzwerkknoten der hauptsächlich Nutzdaten transportiert, aber auch Daten für Managementoperationen befördern kann. Das NE unterscheidet sich von OS und WS darin, daß weder OS noch WS Nutzdaten transportieren.

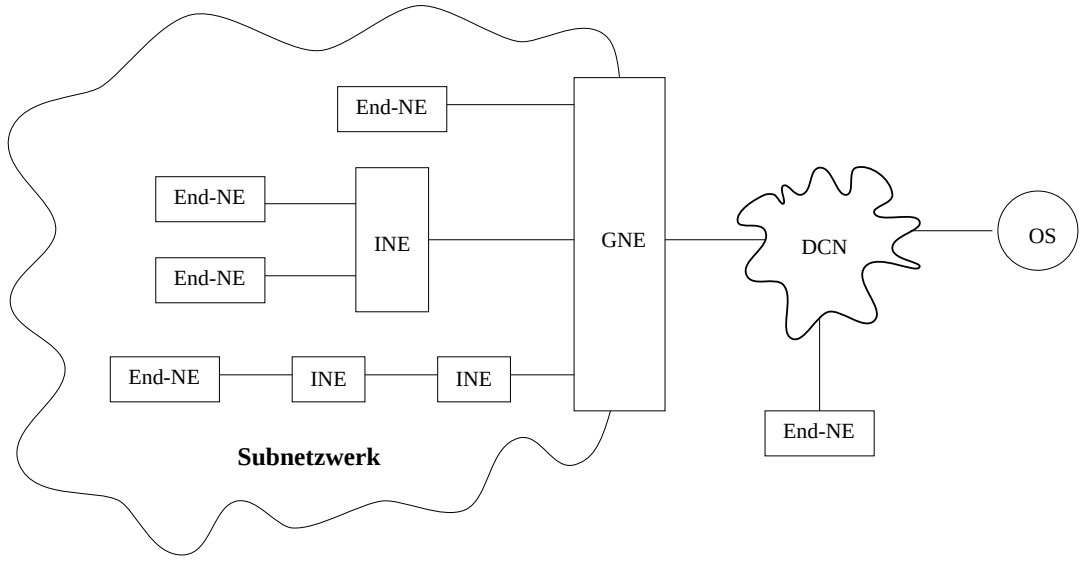

Abbildung 3. Die verschiedenen NE Typen.

Man unterscheidet drei verschiedene Arten von Netzwerkelementen: Gateway NEs (GNE), Intermediate NEs (INE) und End NEs (siehe auch Abb. 3).

Gateway NE: Das GNE hat einerseits direkten Zugriff zu einem paketvermittelnden Netzwerk, einem OS oder einem weiteren GNE und andererseits Zugriff auf ein Subnetzwerk. Es ermöglicht den entfernten Zugriff auf die NEs im Subnetzwerk über EOCs oder OCs. Es routet den Verkehr, der durch den Knoten fließt, bietet statistisches Multiplexen und erfüllt Gateway Funktionen wie Protokollkonvertierung, Nachrichtenumsetzung oder Flußkontrolle für das Subnetzwerk. Ein GNE verhält sich also wie ein Konzentrator oder eine Paketvermittlung.

Intermediate NE: Das INE hat, im Gegensatz zum GNE, keinen direkten Zugriff auf ein paketvermittelndes Netzwerk oder ein OS. Es besitzt aber auch untergeordnete NEs und erfüllt für diese Elemente Routingfunktionen und statistisches Multiplexen. Ein INE kann auch Gateway Funktionen besitzen.

End NE: Das End NE bearbeitet nur seinen eigenen Netzverkehr und braucht daher auch keine Routingfunktionen oder statistisches Multiplexen. Es kann aber Zugriff zu einem beliebigen anderen NE, einem paketvermittelnden Netzwerk oder einem OS haben.

Operations Interface Module: Das *Operations Interface Module* (OIM) stellt eine Menge von generischen Funktionen und Protokollarchitekturen zur Verfügung, die für das Netzwerkmanagement benötigt werden. Die Funktionen und Protokollarchitekturen können in Hardware oder Software oder auch in einer Kombination von beidem implementiert werden. Sie können in einem NE integriert werden, in einem eigenen Gerät untergebracht sein oder auch eine Mischung aus beidem darstellen. Die OIM Funktionen können in zwei Klassen eingeteilt werden – exklusive OIM Funktionen (*OIM dedicated functions* OIM DFs) und gemeinsame OIM Funktionen (*OIM common functions* OIM CFs). OIM DFs werden ausschließlich von NM Anwendungen benutzt, wohingegen die OIM CFs auch mit anderen Anwendungen geteilt werden. OIM CFs sind beispielsweise Routing und Multiplexen, Flußkontrolle oder auch Gateway Funktionen. Ein OIM kann sowohl nur aus OIM DFs bestehen als auch aus einer Auswahl aller OIM Funktionen.

Generische NE Funktionen für NM Data Networking:

Die folgenden NE OIM Funktionen stellen eine kleine Auswahl von Funktionen dar, die für den Betrieb eines TMN bzw. den NM Datentransport von Bedeutung sind. Sie sollen einen Überblick über die Vielfalt der OIM Funktionalität geben.

EOC Access: Diese Funktion ermöglicht einem NE den Zugriff auf einen EOC in einem digitalen Signal. Ein Übertragungsmedium kann viele solcher Signale transportieren, und jedes Signal kann mehrere EOCs beinhalten.

EOC Monitoring: Diese Funktion erlaubt dem NE, den Inhalt eines EOC mitzulesen, ohne seinen Inhalt zu verändern.

EOC Termination: Diese Funktion ermöglicht dem NE, der Endpunkt eines EOCs zu sein und dessen Inhalt zu verarbeiten, d.h. Operationsnachrichten zu analysieren, zu interpretieren und auszuführen, Protokollkonvertierungen durchzuführen und vieles mehr.

EOC Protection Switching: Diese Funktion erlaubt es, beim Ausfall des primären EOC auf den Schutz-EOC auszuweichen. Abhängig von der Architektur des Systems und den Anforderungen an die Kommunikation kann dabei ein EOC mehrere primäre EOCs schützen oder mehrere EOCs können als Schutz für einen einzigen primären EOC vorgesehen sein.

EOC Status Indication and Report: Der EOC kann in drei Zuständen sein: (1) aktiv; (2) „standby" (als Schutz-EOC); und (3) verfügbar. Die NEs sollten den jeweiligen Status der aktiven und der standby EOCs speichern und Änderungen an das entsprechende OS weiterleiten.

EOC Idle Code Insertion: Diese Funktion ermöglicht dem NE, in der Zeit, in der keine Daten im EOC gesendet werden, einen Leerlaufcode zu senden. Ein EOC mit generischer Schnittstelle könnte zum Beispiel das Flag 01111110 von LAPB und LAPD oder auch nur 1en als Leerlaufcode besitzen.

Statistical Multiplexing (Concentration): Diese Funktion macht es dem NE möglich, ankommende Nachrichten zu kombinieren und über den entsprechenden Ausgang weiterzuleiten nachdem evtl. notwendige Maßnahmen wie Protokollkonvertierung oder Pufferung durchgeführt wurden. Die Funktion soll auch zur Kostenreduzierung (Übertragungskosten etc.) beitragen.

Protocol Conversion: Diese Funktion ermöglicht zwei verschiedenen Protokolltürmen miteinander zu kommunizieren. Sie ist nötig, weil NEs, OSs und paketvermittelnde Knoten verschiedene generische Schnittstellen für den NM Datentransport benutzen können.

Address Mapping: Diese Funktion erlaubt einem NE, Nachrichten zwischen zwei unterschiedlichen Schnittstellen zu routen, die zwei verschiedene Adressierungsalgorithmen verwenden oder ein unterschiedliches Adreßformat besitzen.

Message Translation: Diese Funktion wird benötigt, wenn zwei Knoten miteinander kommunizieren wollen, die ein unterschiedliches Nachrichtenformat benutzen. Für manche Funktionen reichen einfache bitorientierte Nachrichten aus. Ein NE kann nun solche Nachrichten interpretieren und einen Bericht mittels zeichenorientierten Nachrichten an das OS senden. Auch kann zum Beispiel der bitorientierte EOC eines kleinen NEs von einem GNE oder INE in einen zeichenorientierten EOC umgewandelt werden.

Routing: Ein NE, das den NM Verkehr seiner untergeordneten NEs sammelt und statistisches Multiplexen durchführt, braucht Routingfunktionen, um den Verkehr der für andere Knoten bestimmt ist, weiterleiten zu können.

Alternate Routing: Alternatives Routen soll die Verläßlichkeit und die Leistung eines NEs steigern. Dadurch, daß alternative Wege zur Verfügung stehen, kann der Ausfall eines Knotens kompensiert werden und somit die Verläßlichkeit erhöht werden. In Überlast- oder Stausituationen kann ein alternativer Weg gewählt werden und damit die Leistung (hier: höherer Durchsatz und niedrigere Paketwartezeit) des Knotens erhöht werden.

Flow Control: Das NE kann ankommende Nachrichten zurückweisen, wenn es zu einem Fehler kommt oder das NE überlastet ist. Dabei kann das NE, je nach Art der Flußkontrolle, alle Nachrichten zurückweisen oder nur bestimmte Nachrichten, die z.B. einen bestimmten Nachrichtenpuffer betreffen.

Message Duplication/Broadcasting: Diese Funktion erlaubt es dem NE, eine Nachricht, die es von einem anderen NE oder einem OS erhalten hat, zu vervielfältigen und an mehrere Ziele weiterzusenden. So können zum Beispiel Nachrichten an alle NEs eines Subnetzwerks gesendet werden oder NEs benachrichtigt werden, die zu einer bestimmten Gruppe gehören.

Die drei NE Arten unterstützen eine unterschiedliche Anzahl von NE Funktionen. Für End NEs hat beispielsweise das Routing und das Multiplexen keine Bedeutung. Den größten Funktionsumfang besitzt das GNE, während ein INE schon mit wesentlich weniger Funktionen auskommen kann.

5 Implementations-Architektur

Um alle erweiterten Dienste und Netzwerkfähigkeiten der heutigen großen Telekommunikationsnetzwerke bereitzustellen, sind eine Vielzahl von Kommunikationswegen zwischen den intelligenten NEs, OSs und WSs notwendig. Es ist aber meist ökonomisch nicht sinnvoll, jedes NE direkt an ein paketvermittelndes Netz (*Packet Switching Network* PSN) anzuschliessen. Besser ist es, wenn ein NE die NM Daten sammelt und dann an das PSN weiterleitet, um Anschlußkosten zu sparen. Weiter helfen generische NE Funktionen und Schnittstellen Kosten zu sparen, da sie Produkte unterschiedlicher Hersteller unterstützen sollen.

Die Kommunikationsfunktionen des TMN werden durch ein Kommunikationsnetzwerk (DCN) realisiert. Da dieses DCN sehr groß werden kann, wird es, wie in Abbildung 4 zu sehen ist, in drei Teile zerlegt. Die Subnetzwerke sind ein *Backbone-DCN* (B-DCN), und zwei Zugriffsnetzwerke die OSs und NEs mit dem B-DCN verbinden. Das erste ist das OS-Zugriffsnetzwerk (*OS-Access Network*) und das zweite das NE-Zugriffsnetzwerk (*NE-Access Network*).

OS-Access Network: Es stellt die Kommunikationswege zwischen den OSs bereit und verbindet die OSs mit dem B-DCN. Die einfachste Architektur besteht aus Punkt-zu-Punkt-Verbindungen zwischen den OSs bzw. zwischen OS und NE. Da diese Architektur meist nicht flexibel und leistungsfähig ist, werden komplexere Architekturen wie Hochgeschwindigkeits-LANs oder FDDI für die Anbindung der OSs an das B-DCN

und für die Kommunikation der OSs untereinander eingesetzt. Die Anbindung an das B-DCN erfolgt auch meist über mehr als ein GNE, um die Zuverlässigkeit und den Durchsatz zu steigern. Natürlich sind weiterhin einzelne direkte Verbindungen zwischen OSs und dem B-DCN oder einem GNE möglich, um speziellen Anwendungen Rechnung zu tragen und die Zuverlässigkeit weiter zu steigern.

NE-Access Network: Es bindet die NEs an das B-DCN an und stellt die Verbindung zu den einzelnen NEs her. Das Netzwerk benutzt EOCs, um die Zugriffskosten auf das B-DCN zu vermindern. Das EOC Konzept wird benutzt, um Netzwerkressourcen (Übertragungsmedien, NEs, Schnittstellen) mit verschiedenen Anwendungen zu teilen um NM Kosten zu reduzieren. Diese Möglichkeit, Ressourcen zwischen Anwendungen aufzuteilen, ist eine der wichtigsten Funktionen der heutigen ISDN Architekturen. Da nicht alle NEs direkt an das B-DCN bzw. ein OS angeschlossen werden können, werden GNEs, INEs und Vermittlungsgeräte benutzt, um ein NE-Access Network zu bauen.

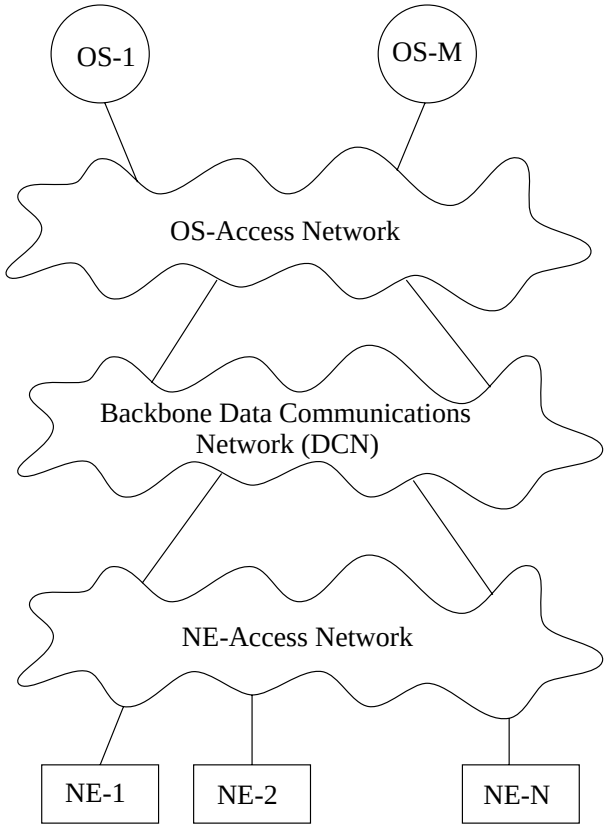

Abbildung 4. Modell einer Implementations Architektur.

Die Vermittlungsgeräte werden eingesetzt, um den NM Netzverkehr derjenigen INEs und End NEs zu sammeln, die nicht an ein GNE angeschlossen sind. Große GNEs, die Vermittlungsfunktionen besitzen, können auch direkt an ein OS angeschlossen sein. Die Übertragungswege und -medien des Telekommunikationsnetzwerks sind schon vorhanden, da die NEs Nutzdaten transportieren. Das Reservieren eines Teils der Kapazität des digitalen Signals für den EOC ermöglicht es, kostengünstig ein NE-

Access Netzwerk zu realisieren.

Backbone-DCN: Das B-DCN kann ein leitungsvermittelndes Netzwerk, ein paketvermittelndes Netzwerk oder ein beliebiges anderes Netzwerk sein. Dieser Abschnitt befaßt sich mit paketvermittelnden Netzwerken als Grundlage für ein B-DCN. Ein eigenständiges PSN wäre keine kosteneffektive Lösung, deshalb werden gemeinsam benutzte PSNs verwendet.

Ein öffentliches PSN kann sowohl NM Daten als auch Nutzdaten transportieren, besitzt aber einige Nachteile. So sind nicht überall Zugänge zum PSN vorhanden und wenn sie vorhanden sind reichen die Übertragungsraten meist nicht aus, so daß Multiplexer/Demultiplexer benötigt werden.

Die GNEs besitzen aber schon OIM Konzentrator- und Routingfunktionen. Ausgewählte GNEs, sogenannte Backbone-GNEs (B-GNEs), können also benutzt werden, um ein eingebettetes PSN zu implementieren. Die Kommunikation mit den B-GNEs wird über EOCs abgewickelt. Die GNEs (oder auch MDs) sammeln und bündeln NM Verkehr aus den untergeordneten NEs und stellen die Funktionen bereit, um zwischen B-DCN und dem NE-Access Network zu kommunizieren. Damit entfällt die Notwendigkeit eines Multiplexers des gemeinsam genutzten PSNs. Die B-GNEs verhalten sich wie einfache Paketvermittlungen, und die GNEs sind Zugriffskonzentratoren. Außerdem hat das Konzept des eingebetteten PSN noch weitere Vorteile: Mit der Aufrüstung eines NEs werden gleichzeitig die Fähigkeiten des TMN aufgerüstet. Das TMN profitiert dann auch von erweiterten Funktionen wie dynamische Bandbreitenreservierung oder Rekonfiguration des Netzwerks. Auch gibt es viele alternative Routing- und Rekonfigurationsmöglichkeiten, da ein GNE meist mit vielen weiteren GNEs verbunden ist.

Ein eingebettetes PSN kann speziell für Netzwerkmanagement ausgelegt sein oder auch mit anderen Anwendungen geteilt werden. Es ist auch möglich, für ein TMN eine Kombination aus verschiedenen PSNs zu verwenden.

6 TMN Schichten und Verteilung von TMN Funktionen

6.1 TMN Schichten

Wenn man ein Managementnetzwerk entwirft, muß man physikalische und logische Gesichtspunkte berücksichtigen. Die physikalischen Gesichtspunkte sind das Management der physikalischen Einheiten (z.B. NEs, Stromversorgung etc.), und die logischen Gesichtspunkte sind logische Einheiten wie Dienste, Netzwerkplanung, Arbeitseinteilung, Accounting, Sicherheit usw. Die TMN Funktionen sind in fünf funktionale Managementschichten aufgeteilt, um den Entwurf eines TMN zu vereinfachen.

Network Element Layer (NEL): Die NEL TMN-Funktionen werden durch die NEs bereitgestellt. Das NEL ist auch hauptsächlich für das Management des NEs verant-

wortlich. Die Funktionen beinhalten so grundlegende Funktionen wie Protokollkonvertierung, Adreßumsetzung, sammeln von Leistungsdaten usw.

Network Element Management Layer (NEML): Das NEML wird oft auch Subnetwork Management Layer genannt. Seine Funktionen werden von einem *Subnetwork Controller* (SNC) zur Verfügung gestellt. Die Schicht verwaltet eine Reihe von NEs (ein Subnetzwerk). Einige Funktionen dieser Schicht sind NML Funktionen, haben aber einen eingeschränkten Wirkungsbereich.

Network Management Layer (NML): Die NML Funktionen werden in der Regel von den OSs erbracht. Diese Funktionen unterstützen die Anwendungen, die eine Ende-zu-Ende Sicht des Telekommunikationsnetzwerks benötigen. Die Schicht sammelt und analysiert die Daten, die sie von der NEML erhält, und erzeugt eine globale Sicht für ihren Bereich. Die Schicht kommuniziert mit anderen Schichten über eine Standardschnittstelle.

Service Management Layer (SML): Die SML ist nicht mit dem Management von physikalischen Einheiten befaßt, sondern stellt den Kontaktpunkt zum Benutzer dar.

Business Management Layer (BML): Sie ist für das ganze Unternehmen verantwortlich. Auf dieser Ebene werden Verträge mit den Operatoren geschlossen und strategische Planungen unterstützt.

6.2 Verteilung von TMN Funktionen

In den heutigen Telekommunikationsnetzwerken ist die Netzwerkmanagement Intelligenz größtenteils in den OSs angesiedelt. Die NEs besitzen so gut wie keine Intelligenz. Da die Telekommunikationsnetzwerke weltweit immer stärker wachsen und immer neue Dienste anbieten, werden die OSs immer mehr mit Funktionen belastet, die sie davon abhalten das Netzwerk effizient zu managen. Deshalb sollten Netzwerkmanagementfunktionen in die bisher praktisch nicht vorhandene NEML Schicht bzw. in die NEL Schicht verlagert werden. In zukünftigen TMN-Architekturen sollten OSFs und MFs gemeinsam von OSs, SNCs und NEs implementiert werden.

Die TMN-Funktionen können dazu in zwei Klassen eingeteilt werden: globale Funktionen und nichtglobale Funktionen. Die nichtglobalen Funktionen brauchen, im Gegensatz zu den globalen Funktionen, keine Ende-zu-Ende Sicht des Netzwerks und können daher von SNCs und NEs übernommen werden. Die globalen Funktionen sollten in den OSs untergebracht sein, da sie eine Ende-zu-Ende Netzwerksicht oder eine sehr weitreichende Kontrolle über das Netzwerk benötigen. Die globalen Funktionen bedienen sich der nichtglobalen Funktionen, um komplexere Aufgaben wie Dienstmanagement, Wiederherstellung des Netzwerks oder dynamische Bandbreitenreservierung zu erledigen.

Um diese Verteilung zu erreichen, werden SNCs benötigt. Die SNCs werden in der Regel von den gleichen Herstellern produziert, die auch die NEs vertreiben. Da SNCs die Managementfunktionen der NEs benutzen und zusätzlich noch verteiltes Management erlauben, ist es möglich, kostengünstige Lösungen zu schaffen. Außerdem können neue Dienste, Produkte und Architekturen schneller eingeführt werden. Die SNCs können auch

helfen, die heutigen Flaschenhälse und Leistungsprobleme zu lösen. Die SNCs können auf der Grundlage von Standard Hard- und Software Plattformen mit Standardschnittstellen entwickelt werden, um Produkte verschiedener Hersteller zu unterstützen und die Anwendungsentwicklung kostengünstig zu gestalten.

7 Standardisierungsbemühungen

ITU-T und ISO

Standards für Telekommunikations-Management-Netzwerke werden von vielen nationalen und internationalen Gremien ausgearbeitet. Hier sollen kurz die wichtigsten Gruppen vorgestellt werden.[1]

Die internationale Standardisierung für das TMN hat die Study Group IV der International Telecommunications Union–Telecommunications (ITU-T) übernommen. Sie befaßt sich mit den allgemeinen Modellen und auch mit Teilen der Dienste und der Technologie. Die Study Group XI trägt zur Standardisierung der Protokolle und des Management-Modells vor allem für SS7 und verschiedene ISDN Anwendungen bei, und die Group XV beschäftigt sich mit den übertragungstechnischen Aspekten der Modelle.

Das ISO/IEC Joint Telecommunications Committee (JTC1) unterstützt die Arbeit der ITU-T vor allem auf dem Gebiet der daten-orientierten Anwendungen. Die ISO befaßt sich mehr mit der technischen Seite, arbeitet aber auch in der ITU-T Study Group VII am Informationsmodell mit.

Die ITU-T Empfehlungen können in diverse Kategorien eingeteilt werden:

- M.3000 Serie: beschäftigt sich mit den allgemeinen Prinzipien des TMN und dem Rahmenwerk,
- M.3100 Serie: hat die TMN Modelle und Objektdefinitionen zum Inhalt,
- M.3200 Serie: beinhaltet die Managementdienste, die ein TMN unterstützen könnte,
- M.3300 Serie: behandelt Aspekte der Arbeitsstationen, die an ein TMN angeschlossen sind,
- M.3400 Serie: befaßt sich mit den Management Funktionen, die TMN Dienste unterstützen.

Für weiter Informationen siehe [Cci89] und [Cci92].

ANSI

In Nord-Amerika entwickelt das Komitee T1 Telekommunikationsstandards für die ANSI. Das technische Unterkomitee T1M1 begann 1986 damit, Schnittstellen-Standards für die TMN Schnittstellen zu entwerfen. Heute ist das Unterkomitee T1M1 damit beschäftigt die TMN Standards der ITU-T an regionale Gegebenheiten anzupassen. Auf Gebieten, auf denen es noch keine Arbeiten der ITU-T gibt oder die Arbeiten nicht mit denen

[1] Stand August 1992

des Komitees übereinstimmen, unterstützt das Komitee die ITU-T durch Informationen über ihre Sichtweise der TMN Modelle.

ANSI Standards sind:

T1.204: Lower Layer Protokolle für OS/NE Schnittstellen
T1.208: Upper Layer Protokolle für OS/NE Schnittstellen
T1.210: Architektur und Funktionen für ein TMN
T1.215: Fehlermanagement-Nachrichten zwischen OSs und NEs
T1.224: Protokolle für OS/OS Schnittstellen

ETSI

Das European Telecommunications Standards Institute (ETSI) ist das europäische Gegenstück zum Komitee T1. Das technische Unterkomitee NA4 ist für die Arbeiten im Zusammenhang mit TMN verantwortlich. Es produziert europäische Standards und stellt auch seine Informationen der ITU-T zur Verfügung. Das ETSI und T1M1 sind dabei, Prozeduren zu überarbeiten, um sowohl den regionalen als auch den internationalen Bedürfnissen Rechnung zu tragen.

ETSI Standards sind beispielsweise:

NA432: Allgemeine Prinzipien des TMN
NA43203: TMN Vokabular
NA433: Modellierung des TMN; Spezifikation von Schnittstellen

TTC

Das TTC ist das japanische Gegenstück zu T1 und ETSI. Das TTC entwirft bisher keine eigenen Standards. Es legt lediglich seine Wünsche der ITU-T zur Prüfung vor.

8 Ausblick

Wir haben hier ein Modell für ein Telekommunikations Management Netzwerk gesehen. Es wurde deutlich, daß für die Verwaltung eines Telekommunikationsnetzwerkes eine Vielzahl von Funktionen nötig ist. Durch die steigende Anzahl von Netzwerkelementen und die immer größeren (globalen) Netzwerke steigt auch der Verwaltungsaufwand. Telekommunikationsnetze dienen auch nicht mehr nur der klassischen Telekommunikation sondern werden auch für den digitalen Datentransport eingesetzt. Durch die immer intelligenteren Netzwerkelemente ist es aber auch möglich und nötig die Verwaltungsaufgaben zu verteilen. So wird ein Netz nicht mehr von einem großen OS verwaltet, sondern die Verwaltungsaufgaben werden aufgeteilt, und das OS hat die einzelnen Aufgaben zu koordinieren. Damit ist das TMN kein eigenständiges Netzwerk mehr, sondern überschneidet sich mit dem zu verwaltenden TN. Das führt dazu, daß Stabilität zu einem entscheidenden Faktor wird, der durch redundante Wege, Subnetzwerke und alternatives Routing gesichert werden soll.

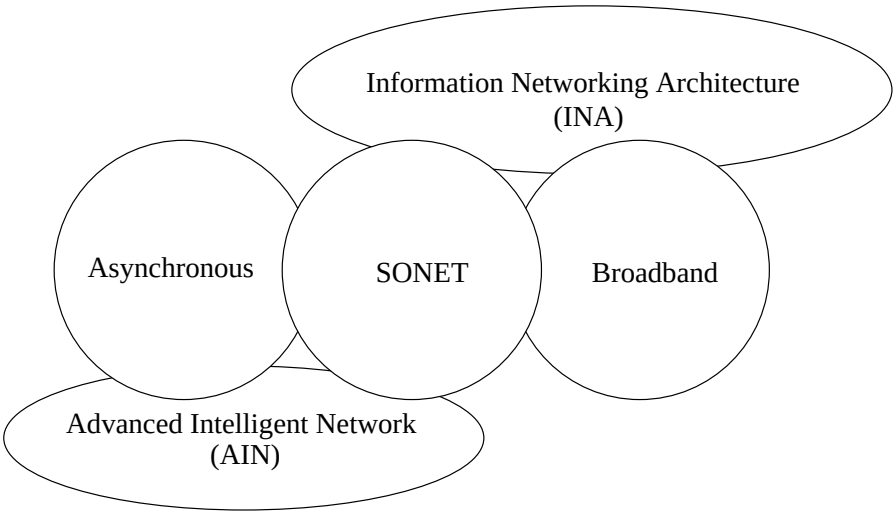

Abbildung 5. Die Entwicklung der TMN Architektur.

Wichtige Designkriterien und Schlüsselfunktionen für zukunftige TMN Netzwerke sind:

- Schnellere Einführung von neuen Diensten und Technologien
- Unterstützung für den Einsatz von Produkten verschiedener Hersteller durch offene Architekturen und Standardschnittstellen
- Wiederverwendbare und Portable Software
- Niedrigere Entwicklungskosten
- Stabilität
- Verteiltes Rechnen
- Objekt-Orientiertes Design
- Modulare Funktionalität

Die heutigen asynchronen Netzwerke werden sich – vor allem durch SONET (Synchronous Optical NETwork) Produkte – zu synchronen Netzwerken wandeln. In den nächsten Jahren werden neue Produkte entwickelt werden, um die neuen Broadband Integrated Services Digital Network (B-ISDN) Dienste zu unterstützen. Insgesamt werden sich die TN-Architekturen zu Information Networking Architectures (INA, s.[Bel94] und Abb. 5) entwickeln.

A Abkürzungsverzeichnis

TN	Telcommunication Network
TMN	Telecommunication Management Network
OS	Operations System
DCN	Data Communication Network
MD	Mediation Device
WS	Workstation
QA	Q Adaptor
NE	Network Element
GNE	Gateway Network Element
INE	Intermediate Network Element
OSF	Operations Systems Function
MF	Mediation Function
DCF	Data Communication Function
NEF	Network Element Funtction
WSF	Workstation Function
QAF	Q Adaptor Function
ECC	Embedded Communications Channel
EOC	Embedded Operations Channel
OC	Operations Channel
SC	Shared Channel
OIM	Operations Interface Module
SNC	Subnetwork Controller
PSN	Packet Switching Network
NEL	Network Element Layer
NEML	Network Element Management Layer
NML	Network Management Layer
SML	Service Management Layer
BML	Business Management Layer

Vergleich aktueller Authentisierungs- und Zertifizierungssysteme

Alexander Will

Kurzfassung

Im folgenden Artikel wird ein Verfahren zur Authentisierung und Privilegienverwaltung in verteilten Systemen von Yoshiki Sameshima vorgestellt, das mit Zertifikaten arbeitet, in denen der geheime Schlüssel des Besitzers eingetragen ist. Im Vergleich zu existierenden Ansätzen und Implementierungen wie Kerberos oder X.509 werden die Vor- und Nachteile bezüglich des Aufwands, der Sicherheit, des Managements und der verwendeten Protokolle aufgezeigt.

1 Einleitung

Authentisierung ist der Nachweis der Identität von Instanzen in einem Kommunikationssystem. Dies ist nötig, um Ressourcen vor Mißbrauch zu schützen. Zertifizierung ist die Zusicherung bestimmter Eigenschaften von Instanzen wie die Identität oder Privilegien. Zertifikate werden zumeist von *vertrauenswürdigen Instanzen* ausgestellt. Beispiele für Zertifikate sind ein Personalausweis, der die Identität bestätigt und eine Bahnfahrkarte, die es erlaubt, Bahn zu fahren. Um in einem elektronischen vernetzen System Zertifikate vor Veränderung oder Fälschung zu schützen, werden Methoden der Verschlüsselung eingesetzt. Verschlüsselung dient zur Sicherung der Kommunikation über unsichere Netze sowie zur Erhaltung der Vertraulichkeit und Integrität von Daten. Ich will nun einige gebräuchliche Verfahren zur Authentisierung vorstellen:

- Anmeldung an einem Rechnersystem mit Benutzernamen und Paßwort. Dies ist im einfachsten Fall ein großes Sicherheitsproblem, da das Paßwort unverschlüsselt über die Leitung bzw. das Netz geschickt wird. Darüber hinaus müssen die Authentisierungsinformationen (in diesem Fall Name und Paßwort) zentral in einer Datenbank gespeichert und verwaltet werden, was auch ein Angriffspunkt darstellt.

- Gegen das Abhören der Kommunikationswege kann man Verschlüsselung mit in jeder Sitzung wechselnden Schlüsseln (session keys) einsetzen. Ein Protokoll, mit dem session keys vergeben werden und mit dem eine Authentisierung erfolgen kann ohne daß ein Paßwort im Klartext übermittelt wird, ist *Kerberos*, das später noch vorgestellt wird.

- Um einen Angriff auf zentral gespeicherte Informationen, wie Listen mit Paßwörtern zu unterbinden kann man Zertifikate einsetzen, die durch entsprechende Verschlüsselung geschützt, öffentlich zugänglich und verteilt gespeichert werden können. Ein solches System wurde von der CCITT mit *X.509* standardisiert. Auch dieses Authentisierungsschema wird später vorgestellt.

– Eine weitere Möglichkeit zur Authentisierung und zur gleichzeitigen Verwaltung von Privilegien und Rechten in verteilten Systemen ist die Verwendung von Zertifikaten, in denen geheime Schlüssel der Benutzer gespeichert sind. Diese *Secret Key Certificates* können durch entsprechende Maßnahmen gesichert auch öffentlich gespeichert werden. Protokolle, die darauf basieren, werden den Schwerpunkt des vorliegenden Textes bilden.

2 Verschlüsselungsverfahren

Die besten Verfahren zur Authentisierung und Zertifizierung sind nur so gut wie die eingesetzten Verschlüsselungsverfahren. Die public key Verschlüsselung wird i.A. als stärker als symmetrische Verfahren angesehen, da bei ihr längere Schlüssel verwendet werden. Gerade bei Verwendung von symmetrischer Verschlüsselung und session keys sollte deshalb auf eine ausreichende Schlüssellänge geachtet werden.

2.1 Symmetrische Verfahren

Bei einer symmetrischen Verschlüsselung teilen sich zwei Kommunikationspartner einen gemeinsamen Schlüssel, der zum Ver- bzw. Entschlüsseln von Nachrichten dient. Ein Beispiel für ein symmetrisches Verfahren ist *DES*. Bild 6 zeigt die Anwendung symmetrischer Verfahren.

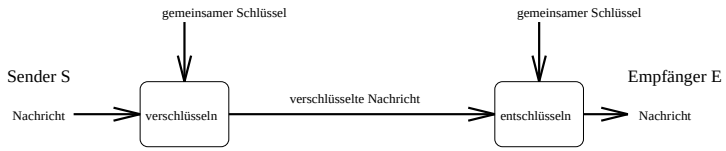

Abbildung 6. Symmetrische Verschlüsselung

2.2 Public Key Verschlüsselung

Bei asymmetrischen (public key) Verschlüsselungsverfahren besitzt jeder Teilnehmer der Kommunikation ein Schlüsselpaar, bestehend aus einem für alle zugänglichen öffentlichen Schlüssel und einem geheimen Schlüssel. Will nun der Sender dem Empfänger eine Nachricht schicken, so *verschlüsselt* er sie mit dessen öffentlichem Schlüssel. Nur der rechtmäßige Empfänger kann sie dann wieder entschlüsseln. Bild 7 zeigt die Ver- und Entschlüsselung bei public key Verfahren.

Umgehehrt kann man aber auch die Nachricht mit dem eigenen geheimen Schlüssel verschlüsseln, so kann der Empfänger, der sie mit dem öffentlichen Schlüssel des Senders dekodiert, sicher sein, daß die Nachricht nur vom Sender kommen kann. Diesen Vorgang, wie er in Bild 8 dargestellt wird, nennt man auch *unterschreiben*.

Ein Verfahren, das diese Eigenschaften besitzt ist *RSA*. Es gilt bei hinreichender Schlüssellänge heute als sicher. Ein Nachteil der asymmetrischen Verfahren ist meist die Ver-

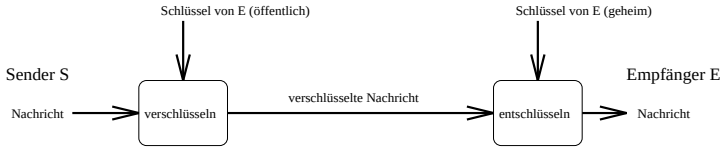

Abbildung 7. Asymmetrische Verschlüsselung

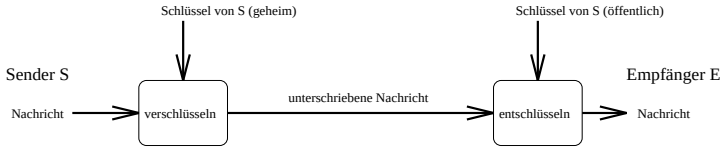

Abbildung 8. Unterschreiben einer Nachricht

bzw. Entschlüsselungsgeschwindigkeit. Während man symmetrische Verfahren wie DES meist in Hardware implementieren kann, (damit lassen sich dann Massendaten, wie sie zum Beispiel bei der Audioübertragung anfallen in Echtzeit verschlüsseln) benötigen public key Verfahren wie RSA oft Langzahlarithmetik, was die Geschwindigkeit einschränkt.

3 Kerberos

Kerberos ist eine Sicherheitssoftware für verteilte Systeme, die am MIT entwickelt wurde. Es hat sich zum de facto Standard zur Authentisierung in vernetzten Client-Server Umgebungen entwickelt [Sta94]. Es basiert auf symmetrischer Verschlüsselung, um über das Netz verbreitete Nachrichten zu sichern. Darüber hinaus setzt es Zeitlimits ein, um die Gefahr von unberechtigter Nutzung von Zufriffsprivilegien zu minimieren. Kerberos verläßt sich auf die Authentisierung durch eine "vertrauenswürdige dritte Instanz". Das bedeutet, das alle Benutzer und Diensterbringer einem ausgezeichneten Authentisierungsdienst vertrauen. Der Authentisierungsdienst speichert die Paßwörter aller Benutzer und aller anderen Server des Systems in einer zentralen Datenbank. Bild 9 zeigt die in einem Kerberos System vorkommenden Instanzen.

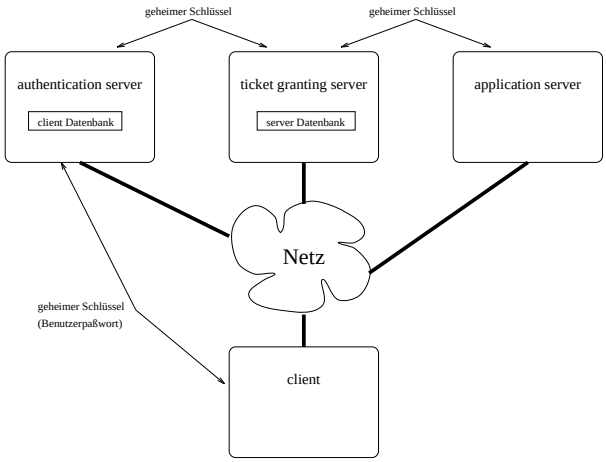

Abbildung 9. Architektur eines Kerberos Systems

Der Benutzer schickt dem *authentication server* eine Nachricht, die die User ID des Benutzers und eine Anfrage nach einem sogenannten *ticket granting ticket* beinhaltet.

Der *authentication server* antwortet mit einem *ticket granting ticket* und einem session key. (Siehe Bild 10) Die gesamte Nachricht wird mit dem Paßwort des Benutzers verschlüsselt. Nachdem die Antwort des *authentication server* auf der Workstation des Benutzers angekommen ist, wird dessen Paßwort abgefragt und damit die Antwort entschlüsselt. Das *ticket granting ticket* besteht aus der ID des ticket granting servers, der ID des Benutzers, einem Zeitstempel, einer Gültigkeitsdauer und einer Kopie des session keys. Das gesammte *ticket granting ticket* ist mit einem Schlüssel verschlüsselt, den der *authentication server* und der *ticket granting server* miteinander teilen, damit kein Benutzer das *ticket granting ticket* ändern kann.

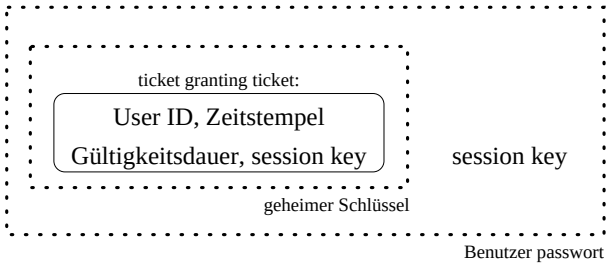

Abbildung 10. Antwort des authentication servers

Der Benutzer erzeugt nun einen sogenannten Authenticator, der aus der ID des Benutzers, dessen Adresse und einem weiteren Zeitstempel besteht. Der Benutzer schickt eine Nachricht bestehend aus dem Authenticator, dem *ticket granting ticket* und einer Anfrage nach dem gewünschten Server (z.B.: einem Datei- , Applikations- oder Druckdienst) an den *ticket granting server*. Der Authenticator wird dabei mit dem session key, der auch im *ticket granting ticket* steht verschlüsselt.

Der *ticket granting server* überprüft die ID des Benutzers mit der ID im *ticket granting ticket*, die Adresse des Benutzers mit der Quelladresse der Nachricht und den Zeitstempel.

Ist alles in Ordnung, so antwortet der *ticket granting server* mit einem neuen session key und einem *service granting ticket*, das unter anderem die Benutzer ID und den neuen session key enthält. Das *service granting ticket* ist mit einem Schlüssel verschlüsselt, den der *ticket granting server* und der Server, dessen Dienst der Benutzer in Anspruch nehmen will, teilen. Die ganze Antwort ist, wie Bild 11 zeigt, mit dem alten session key des Benutzers verschlüsselt.

Zu diesem Zeitpunkt kann der Benutzer auch die Authentität des Servers überprüfen. Er schickt ihm einen Authenticator, verschlüsselt mit dem (neuen) session key und das *service granting ticket*. Der Server entschlüsselt den Authenticator mit dem session key im *service granting ticket* und antwortet mit dem Zeistempel des Authenticators nachdem er ihn um 1 erhöht hat.

Nach diesem Prozeß haben sich sowohl Benutzer als auch Server gegenseitg authentisiert und teilen einen gemeinsamen session key, mit dem sie ihre Nachrichten verschlüsseln.

Das Kerberos System hat den Vorteil, daß kein Paßwort unverschlüsselt übertragen wird.

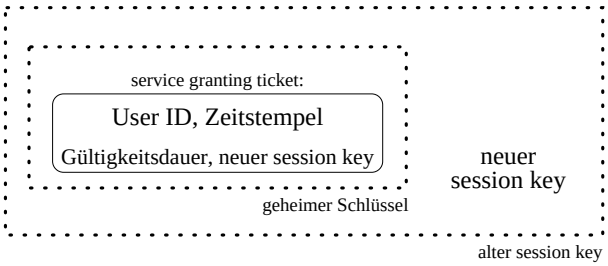

Abbildung 11. Antwort des ticket granting servers

Das wird aber dadurch erkauft, daß alle Benutzerinformationen zentral gespeichert sind. Dies hat zur Folge, daß der Rechner, auf dem der *authentication server* läuft physikalisch vor Zugriff geschützt werden muss. Wird die Verfügbarkeit der Server durch Replikation erreicht, so bietet das notwendige Management zur Erhaltung der Konsistenz der einzelnen Datenbanken einen Angriffspunkt, um in das System einzubrechen. Wegen der zentralen Speicherung der Benutzerinformationen eignet sich Kerberos nur für kleinere Domänen. Ein Authentisierungsverfahren, das weltweit arbeiten soll, muß dagegen dezentral aufgebaut sein, damit nicht alle Verantwortung für das System und dessen Verfügbarkeit an einem Punkt liegt.

4 X.509

Die public key Verschlüsselung bietet eine andere Möglichkeit zur Authentisierung. Wenn ich den public key meines Kommunikationspartners kenne, so kann ich eine sichere Kommunikation aufbauen. Die Frage ist nur noch: Wie bekomme ich den public key, ohne daß dieser von Dritten gefälscht werden kann?

Ein Ansatz zur Authentisierung ist die Benutzung von Zertifikaten. Ein Zertifikat beschreibt eine Eigenschaft (z.B.: Identität oder Recht) eines Benutzers und wird von einer vertrauenswürdigen Instanz ausgestellt. Wie Bild 12 zeigt, enthalten Zertifikate nach dem X.509 Standard eine Versionsnummer, eine laufende Seriennummer, die ausstellende Instanz, einen Gültigkeitsbereich, das zertifizierte Objekt und dessen public key. Das Ganze ist vom Aussteller unterschrieben. [Sta95]

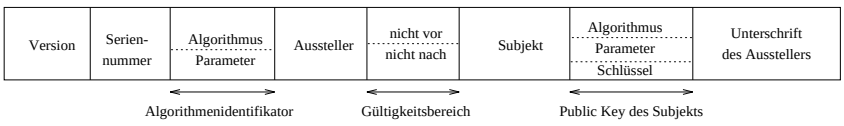

Abbildung 12. X.509 Zertifikat

Der Standard benutzt folgende Notation um ein Zertifikat zu definieren:
$\mathbf{Y}\ll\mathbf{X}\gg$ = Zertifikat des Benutzers X ausgestellt durch Y
$\mathbf{Y}\{\mathbf{I}\}$ = I unterschrieben von Y. Es besteht aus I, dem ein verschlüsselter Hashwert über I angehängt wurde.

Zertifikate haben folgende Eigenschaften:

– Jeder Benutzer mit dem public key des Ausstellers kann die zertifizierten public keys lesen.

– Niemand außer dem Aussteller kann die Zertifikate verändern, ohne daß dies entdeckt wird.

Aus diesen Gründen können Zertifikate ohne besonderen Schutz in öffentlich zugänglichen Verzeichnissen gespeichert werden.

In großen Systemen ist es praktisch unmöglich,daß alle Benutzer den selben Aussteller von Zertifikaten benutzen. Wenn sich aber die Aussteller gegenseitig zertifizieren, kann eine *Kette des Vertrauens* zwischen den Benutzern aller Aussteller geschaffen werden. Das funktioniert folgendermaßen:
Angenommen Benutzer A hat ein Zertifikat von Aussteller X_1 und Benutzer B eins von Aussteller X_2. Darüber hinaus gibt es ein Zertifikat von X_2 ausgestellt von X_1. Dann kann A den public key von X_2 lesen und damit B's Zertifikat ausgestellt von X_2 überprüfen. In der X.509 Notation wird diese Kette folgendermaßen dargestellt:
$X_1 \ll X_2 \gg X_2 \ll \mathrm{B} \gg$
Dieses Schema ist nicht auf zwei Zertifikate beschränkt. X.509 schlägt vor,daß die Aussteller von Zertifikaten hierarchisch angeordnet sind.

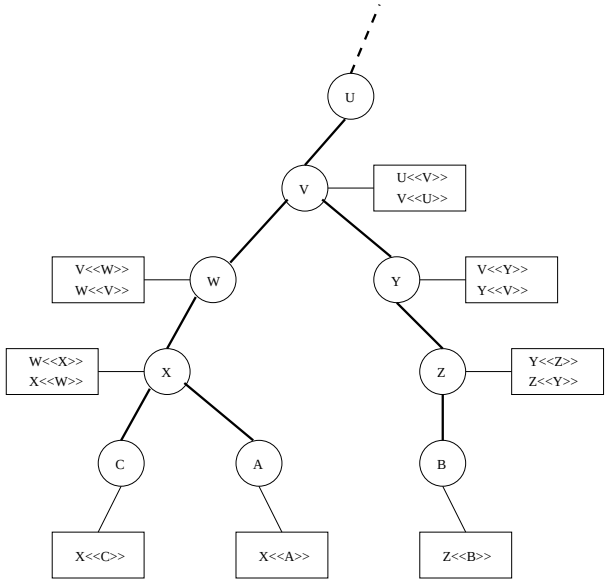

Abbildung 13. Beispiel für hierarchische Anordnung von Ausstellern von Zertifikaten

In dem in Bild 13 gezeigten Beispiel kann A durch
$\mathrm{X} \ll \mathrm{W} \gg \mathrm{W} \ll \mathrm{V} \gg \mathrm{V} \ll \mathrm{Y} \gg \mathrm{Y} \ll \mathrm{Z} \gg \mathrm{Z} \ll \mathrm{B} \gg$
eine Kette zu B aufbauen.

Zertifikate können vor Ablauf der Gültigkeitsdauer für ungültig erklärt werden, indem der Aussteller eine Liste führt, in der alle zurückgenommenen Zertifikate stehen. Selbstverständlich ist diese Liste vom Aussteller unterschrieben.

Aufbauend auf den public key Zertifikaten definiert X.509, wie in Bild 14 gezeigt, drei Alternativen zur Authentisierung:

– Ein-Weg Authentisierung:
  Ein-Weg Authentisierung bei einer Nachricht von Benutzer A zu Benutzer B sichert folgendes:

  1. Die Identität von A und daß die Nachricht von A erzeugt wurde.

  2. Das Ziel der Nachricht war B.

  3. Die Integrität und Orginalität (sie wurde nur einmal gesendet) der Nachricht.

  Die Nachricht besteht aus: einem Zeitstempel t_A, einem Zähler r_A und der Identität von B. Sie wird mit A's secret key unterschrieben. Der Zähler r_A ist während des Gültigkeitszeitraums der Nachricht eindeutig, so verhindert man Wiederholungsattacken. Zusätzlich können noch Daten *sgnData* und ein sesion key K_{AB} übermittelt werden, wobei der session key noch mir B's public key verschlüsselt wird.

– Zwei-Wege Authentisierung:
  Schickt B eine gleichartige Nachricht an A zurück mit t_B, r_B und A, so kennen beide Seiten die Identität der anderen.

– Drei-Wege Authentisierung:
  Schickt nun A zusätzlich den Zähler r_B zurück, so müssen die Zeitstempel nicht mehr geprüft werden, da jede Seite den Zähler der anderen Seite zurückgeschickt hat. So können Wiederholungsattacken verhindert werden. Dieses Verfahren wird eingesetzt, wenn es keine synchronisierte Uhrzeit im System gibt.

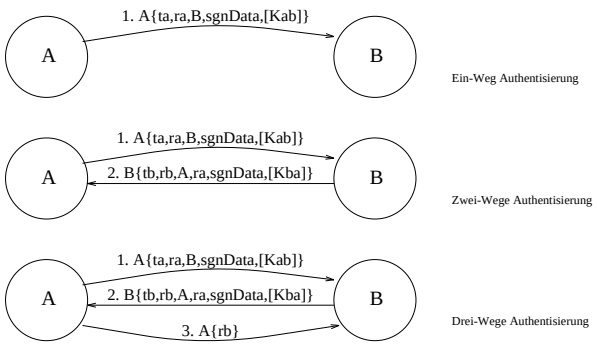

Abbildung 14. X.509 Authentisierungsverfahren

Zusammenfassend ist zu sagen, daß sich durch die Verteilung der Zertifikate und durch die Möglichkeit, die Authentisierung selbst durchzuführen, X.509 gut für die Authentisierung in sehr großen Netzen wie dem Internet eignet, denn die Zertifikate werden ja nur im Verzeichnisdienst *gespeichert* und die Ausstellung der Zertifikate kann *offline* vorgenommen werden. Andererseits bergen lange "Ketten des Vertrauens" immer das Risiko, daß ein schwaches Glied existiert, das angreifbar ist.

5 Secret Key Zertifikate

In den folgenden Protokollbeschreibungen wird der *Authentisierungsserver* AS und der *Privilegserver* PAS genannt. Ein secret key Zertifikat [SMSP96] beinhaltet den secret key des Benutzers, verschlüsselt mit dem secret key des AS oder PAS. Die Struktur ist ähnlich zu den in X.509 definierten public key Zertifikaten. Das Zertifikat eines Benutzers X mit dessen secret key K_X verschlüsselt mit dem Schlüssel K_S des Servers S wird $\{X, \{K_X\}_{K_S}, L\}^{K_S}$ geschrieben, wobei $\{I\}^K$ Information I unterschrieben mit Schlüssel K bedeutet und $\{I\}_K$ für Information I verschlüsselt mit K steht. L enthält dabei Kontrollinformationen des Zertifikats.

5.1 Einfaches Protokoll

Dieses Protokoll zeigt, wie der Benutzer C einen session key vom AS zum PAS erhält, dann einen session key vom PAS zum Anwendungsserver S und damit Zugriff auf S mit dem Recht P. Das Ticket $T^K_{X,Y}$ ist ähnlich wie in Kerberos definiert: $\{X, Y, K_{X,Y}, l, t_s\}_K$ wobei l die Gültigkeitsdauer und t_s der Zeitstempel ist. Das Zertifikat $\{X, \{K_X\}_{K_S}, L\}^{K_S}$ wird $KeyCert_{S,X}$ und $\{X, P, L\}_{K_S}$ wird $PrivCert_{S,X,P}$ genannt. Bild 15 beschreibt die einzelnen Schritte des Protokolls.

1. **Authentisierungsanforderung:** C fordert vom AS ein Ticket für den PAS. Er sendet dazu die Schlüsselzertifikate von C und PAS.

$$C -> AS : C, PAS, KeyCert_{AS,C}, KeyCert_{AS,PAS}$$

2. **Authentisierungsantwort:** AS überprüft die Schlüsselzertifikate, liest K_C und K_{PAS} und erzeugt einen session key $K_{C,PAS}$. Dann erzeugt AS Tickets für C und PAS mit dem Benutzernamen, dem Namen des PAS, dem erzeugten session key und einem Zeitstempel und verschlüsselt sie mit K_C bzw. K_{PAS}. Diese sendet er zurück zu C.

$$AS -> C : T^{K_C}_{C,PAS}, T^{K_{PAS}}_{C,PAS}$$

3. **Privileganforderung:** C berechnet $K_{C,PAS}$ mit seinem secret key und erzeugt eine Privileganforderung mit dem Servernamen S, einem Privileg P zum Zugriff auf S und dem Zeitstempel aus dem Ticket. Diese verschlüsselt er mit $K_{C,PAS}$ und schickt es zusammen mit dem Ticket für PAS, dem Pivilegzertifikat des Privilegs P und dem Schlüsselzertifikat von S ab.

$$C -> PAS : T^{K_{PAS}}_{C,PAS}, \{S, P, l, t_s\}_{K_{C,PAS}}, PrivCert_{PAS,C,P}, KeyCert_{PAS,S}$$

4. **Privilegantwort:** PAS berechnet $K_{C,PAS}$ aus $T^{K_{PAS}}_{C,PAS}$, entschlüsselt damit die Anforderung und vergleich die Zeitstempel der Anforderung und des Tickets. Dann berechnet er Tickets für C und S mit dem Benutzernamen C, dem Servernamen C, einem neuen session key und einem neuen Zeitstempel und verschlüsselt sie mit $K_{C,PAS}$ und

K_S aus dem Schlüsselzertifikat von S. Darüber hinaus erzeugt PAS ein temporäres Privilegzertifikat, das bestätigt, daß C das Privileg P hat, und antwortet damit und mit den beiden Tickets.

$$PAS -> C : T_{C,S}^{K_{C,PAS}}, T_{C,S}^{K_S}, \{C, P, l\}_{K_S}$$

5. **Serveranforderung:** Der Benutzer berechnet $K_{C,S}$ aus $T_{C,S}^{K_{C,PAS}}$ und erzeugt die Anfrage mit dem Servernamen S und dem Zeitstempel aus dem Ticket. Die Anfrage wird mit dem session key verschlüsselt und zusammen mit Ticket für S und dem temporären Privilegzertifikat zu S abgeschickt.

$$C -> S : T_{C,S}^{K_S}, \{S, l, t_s\}_{K_{C,S}}, \{C, P, l\}_{K_S}$$

6. **Serverantwort:** S bekommt den session key aus dem Ticket, entschlüsselt das temporäre Privilegzertifikat und die Anfrage und überprüft, daß C das Privileg P hat. Dann antwortet er mit dem um eins erhöhten Zeitstempel.

$$S -> C : \{t_S + 1\}_{K_{C,S}}$$

7. **Authentisierung:** C entschlüsselt die Antwort, überprüft den Zeitstempel und vervollständigt damit die Authentisierung. Jetzt teilen C und S den session key, mit dem die weitere Kommunikation verschlüsselt wird.

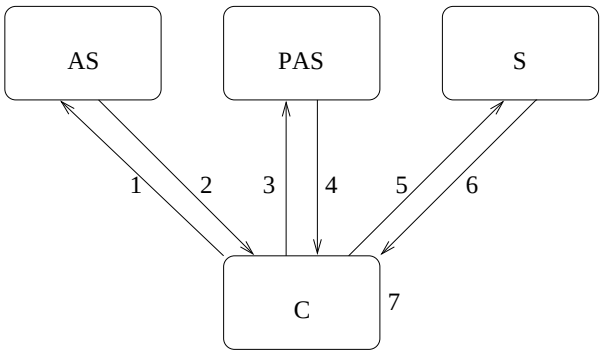

Abbildung 15. Aufrufreihenfolge beim einfachen Protokoll

5.2 Erweitertes Protokoll

Beim einfachen Protokoll benennt der Benutzer den gewünschten Server explizit. Manchmal ist es aber wünschenswert, nur den Diensttyp ST zu spezifizieren. Das erweiterte Protokoll ermöglicht dies. Bild 16 zeigt die Aufrufreihenfolge des erweiterten Protokolls.

1. **Privileganforderung:** Wir nehmen an, daß der Benutzer C sich gegenüber AS authentisiert und das Ticket für PAS erhalten hat. C sendet die Privileganforderung für

ST genau wie im einfachen Protokoll, nur ohne Schlüsselzertifikat für einen bestimmten Server.

$$C -> PAS : T_{C,PAS}^{K_{PAS}}, \{ST, P, l, t_s\}_{K_{C,PAS}}, PrivCert_{PAS,C,P}$$

2. **Privilegantwort:** PAS erzeugt einen session key $K_{ST,PAS}$ und zwei Tickets, die er mit $K_{C,PAS}$ und $K_{ST,PAS}$ verschlüsselt. Dann erzeugt er ein temporäres Privilegzertifikat, in dem steht, daß C das Privileg P hat. Das Privvilegzertifikat wird mit $K_{ST,PAS}$ verschlüsselt. Darüber hinaus erzeugt er ein Ticket mit $K_{ST,PAS}$ für sich selbst. Alle drei Tickets und das temporäre Privilegzertifikat werden an C geschickt.

$$PAS -> C : T_{C,ST}^{K_{C,PAS}}, T_{C,ST}^{K_{ST,PAS}}, T_{ST,PAS}^{K_{PAS}}, \{C, P, l\}_{K_{ST,PAS}}$$

3. **Dienstanforderung:** Der Benutzer richtet eine Dienstanforderung an einen Server, der ST bereitstellt. Die Anforderung ist die gleiche, wie im einfachen Protokoll, außer daß $T_{ST,PAS}^{K_{PAS}}$, $T_{C,PAS}^{K_{PAS}}$ und der Diensttyp mitgeschickt werden.

$$C -> S : T_{C,ST}^{K_{ST,PAS}}, \{ST, l, t_s\}_{K_{C,ST}}, \{C, P, l\}_{K_{ST,PAS}}, T_{ST,PAS}^{K_{PAS}}, T_{C,PAS}^{K_{PAS}}, ST$$

4. **Zertifikatanforderung:** Der Server S, der die Anforderung bekommt, sendet eine Zertifikatanforderung an PAS und benutzt dabei $K_{S,PAS}$ aus der Authentisierung. Er sendet das Privilegzertifikat, in dem steht, daß S ST unterstützt, sowie alles, was er vom Benutzer bekommen hat, bis auf ST.

$$S -> PAS : T_{S,PAS}^{K_{PAS}}, \{PAS, l, T_s\}_{K_{S,PAS}}, PrivCert_{PAS,S,ST},$$

$$T_{C,ST}^{K_{ST,PAS}}, \{ST, l, t_s\}_{K_{C,ST}}, \{C, P, l\}_{K_{ST,PAS}}, T_{ST,PAS}^{K_{PAS}}, T_{C,PAS}^{K_{PAS}}$$

5. **Zertifikantwort:** PAS bekommt $K_{ST,PAS}$ mit $T_{ST,PAS}^{K_{PAS}}$, entschlüsselt $T_{C,ST}^{K_{ST,PAS}}$ und erfährt dann, daß C auf einen Server, der ST bereitstellt, zugreift und zwar mit dem session key $K_{C,ST}$. Die Überprüfung der Privileg-Zertifikate zeigt, daß S ST unterstützt und daß C das Privileg P hat. Dann antwortet PAS dem Server mit dem Benutzer C, dessen Privileg P, einem neuen session key zwischen C und S, dem Zeitstempel und Gültigkeitsbereich aus der Anfrage von C. PAS verschlüsselt diese Informationen mit $K_{S,PAS}$ und schickt auch ein Ticket, das den session key zwischen C und S enthält.

$$PAS -> S : T_{C,S}^{K_{C,PAS}}, \{C, P, K_{C,S}, l, t_s\}_{K_{S,PAS}}$$

6. **Dienstantwort:** S entschlüsselt die Antwort um den zugreifenden Benutzer und dessen Privileg zu überprüfen und schickt die Dienstantwort mit dem neuen Ticket zurück.

$$S -> C : T_{C,S}^{K_{C,PAS}}, \{t_s + 1\}_{k_{C,S}}$$

7. **Authentisierung:** C entschlüsselt das neue Ticket mit $K_{C,PAS}$, bekommt damit den neuen sessoin key $K_{C,S}$ und dessen Zeitstempel und authentisiert den Server durch die Berechnung von $t_s + 1$.

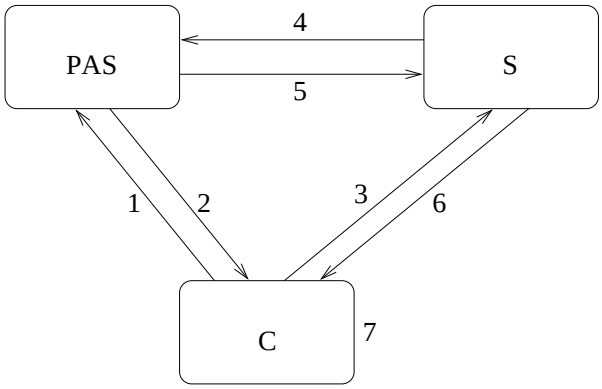

Abbildung 16. Aufrufreihenfolge beim erweiterten Protokoll

5.3 Weitergabe von Privilegien

Die Weitergabe von Privilegien wird gebraucht, wenn ein Benutzer ein Zwischensystem beauftragt, in dessen Namen zu handeln. Ein Beispiel dafür ist ein Druckdienst: Der Benutzer beauftragt den Druckdienst eine Datei von einem Fileserver zu drucken. In diesem Fall braucht der Druckdienst Leserechte auf die Datei, die er normalerweise nicht hat, also durch den Benutzer zugewiesen werden sollte. Im folgenden Protokoll beauftragt ein Benutzer C ein System I mit einem Job, der das Privileg P erfordert mit der Einschränkung R. Bild 17 zeigt die teilnehmenden Instanzen.

1. **Delegierungsanforderung:** C erbittet von PAS die Erzeugung eines Privilegs. Die Anfrage beinhaltet das Zwischensystem, das Privileg und die Einschränkung und wird mit $K_{C,PAS}$ verschlüsselt. Der Benutzer schickt darüberhinaus noch das Schlüsselzertifikat von I und das Privileg-Zertifikat, das beweist, daß er das Privileg P besitzt.

$$C -> PAS : T_{C,PAS}^{K_{PAS}}, \{I, P, R, l, t_s\}_{K_{C,PAS}}, PrivCert_{PAS,C,P}, KeyCert_{PAS,I}$$

2. **Delegierungsantwort:** PAS entschlüsselt die Anforderung mit dem session key aus $T_{C,PAS}^{K_{PAS}}$ und erfährt, daß C Delegierung von P mit R zu I im Zeitraum l wünscht. Das Privilegzertifikat bestätigt, daß C das Privileg P hat. PAS antwortet mit dem Namen des Zwischensystems I, dem Privileg P, dessen Einschränkung R, der Gültigkeitszeit l und dem Namen des Benutzers C, der zur Abrechnung benutzt werden kann. Das erzeugte Privileg wird mit dem Schlüssel des PAS verschlüsselt und zusammen mit Tickets für C und I zurückgeschickt.

$$PAS -> C : T_{C,I}^{K_{C,PAS}}, T_{C,I}^{K_I}, \{I, P, R, l, \{C\}\}_{K_{PAS}}$$

3. **Dienstanforderung:** C schickt I eine Anforderung zusammen mit dem Privileg, dem Ticket für I, P und R

$$C -> I : T^{K_I}_{C,I}, \{I, l, t_s\}_{K_{C,I}}, \{I, P, R, l, \{C\}\}_{K_{PAS}}, P, R$$

4. **Dienstantwort:** I schickt eine Antwort zur Authentisierung.

$$I -> C : \{t_s + 1\}_{K_{C,I}}$$

5. **Privileganforderung:** I schickt PAS eine Privileganforderung mit dem erhaltenen Privileg und dem Schlüsselzertifikat des Zielservers S. Die Anfrage beinhaltet S, das Privileg P, die Einschränkung, die Gültigkeitszeit und den Zeitstempel des Tickets und wird verschlüsselt mit dem session key zwichen I und PAS.

$$I -> PAS : T^{K_{PAS}}_{I,PAS}, \{S, P, R, l, t_s\}_{K_{I,PAS}}, \{I, P, R, l, \{C\}\}_{K_{PAS}}, KeyCert_{PAS,S}$$

6. **Privilegantwort:** PAS entschlüsselt die Anforderung mit dem session key aus $T^{K_{PAS}}_{I,PAS}$ und überprüft die Privilegien. PAS antwortet mit Tickets für I und S und einem neuen Privileg, das P,R,l und {C,I} enthält. Die Komponente {C,I} wird zur Abrechnung benutzt und besagt, daß das Privileg von C stammt, über das System I.

$$PAS -> I : T^{K_{I,PAS}}_{I,S}, T^{K_S}_{I,S}, \{S, P, R, l, \{C, I\}\}_{K_S}$$

7. **Dienstanforderung:** I schickt eine Anforderung mit dem neuen Privileg.

$$I -> S : T^{K_S}_{I,S}, \{S, l, t_s\}_{K_{I,S}}, \{S, P, R, l, \{C, I\}\}_{K_S}$$

8. **Dienstantwort:** S schickt eine Antwort zur Authentisierung.

$$S -> I : \{t_s + 1\}_{K_{I,S}}$$

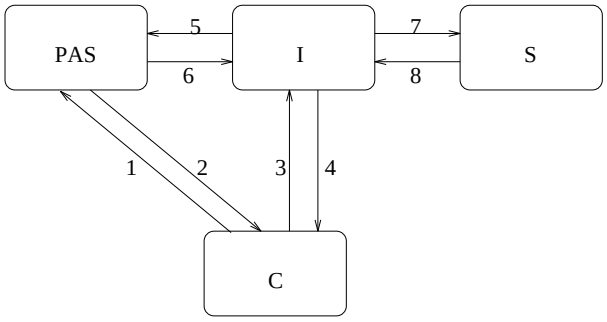

Abbildung 17. Aufrufreihenfolge bei der Weitergabe von Privilegien

5.4 Nachrichtensysteme

In Systemen, die Nachichten austauschen, hat die Verwendung der Zertifikate zwei Vorteile: Der eine ist die Authentisierung des Privilegs des Senders und dessen Kontext. Der andere ist die Auswahl des Empfängers durch dessen Privilegien und Kontext.

5.4.1 Privileg- und Kontextauthentisierung Der erste Vorteil der Anwendung von Zertifikaten ist die Authentisierung der Privilegien und Kontextinformationen, nicht nur des Namens sondern auch dessen Rolle und Titel sowie Kontextinformation wie Sendezeitpunkt, Sendeort oder Nachrichtentyp durch den Empfänger. Im folgenden Protokoll bittet der Sender O mit Privileg P PAS das Privileg und den Kontext C zu zertifizieren. Bild 18 beschreibt die Aufrufreihenfolge.

1. **Authentisierungsanforderung:** O erzeugt die Nachricht MSG und berechnet einen Hashwert h. Dieser Wert wird zusammen mit P und C mit dem Schlüssel von O verschlüsselt und zusammen mit dem Schlüsselzertifikat und dem Privilegzertifikat von O zum PAS geschickt. Der Kontext C ist entweder ein Kontexttyp wie Zeit, Ort oder ein Kontextattribut (Typ plus Wert) wie Nachrichtentyp.

$$O -> PAS : \{h, P, C\}_{K_O}, KeyCert_{PAS,O}, PrivCert_{PAS,O,P}$$

2. **Authentisierungsantwort:** PAS entschlüsselt die Anforderung mit dem Schlüssel des Senders aus dessen Schlüsselzertifikat. PAS überprüft, daß O das Privileg P hat und erzeugt Authentisierungsinformationen bestehend aus dem Hashwert h, dem Sender, dem Privileg P und dem Kontext C, verschlüsselt mit dem Schlüssel des PAS. C wird je nach Typ unterschiedlich behandelt: War C in der Anfrage ein Typ, so besteht C in der Antwort aus dem Typ und dem dazugehörigen Wert. (z.B. war C in der Anfrage gleich *time* so steht im Ergebnis *time = 1.1.1999*) War C in der Anfrage ein Attribut, so wird das Attribut selbst zurückgegeben. z.B. $messageType = PostScript$

$$PAS -> O : \{h, O, P, C\}_{K_{PAS}}$$

3. **Abschicken der Nachricht:** O sendet die Nachricht MSG zusammen mit den Authentisierungsinformationen an den Empfänger R.

$$O -> R : MSG, \{h, O, P, C\}_{K_{PAS}}$$

4. **Zertifizierungsanforderung:** Der Empfänger R sendet diee Authentisierungsinformationen zusammen mit seinem Schlüsselzertifikat an den PAS.

$$R -> PAS : \{h, O, P, C\}_{K_{PAS}}, KeyCert_{PAS,R}$$

5. **Zertifizierungsantwort:** PAS entschlüsselt die Anforderung und verschlüsselt sie wieder mit dem Schlüssel von R aus dessen Schlüsselzertifikat.

$$PAS -> R : \{h, O, P, C\}_{K_R}$$

6. **Überprüfung der Nachricht:** R berechnet den Hashwert von MSG und vergleicht ihn mit dem Hashwert der Antwort. Sind beide gleich, so ist die Integrität der Nachricht, sowie die Indentität des Senders, seine Privilegien sowie der Senderkontext gesichert.

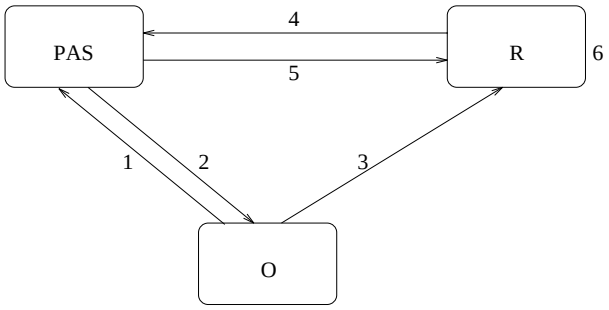

Abbildung 18. Aufrufreihenfolge bei Privileg- und Kontextauthentisierung

5.4.2 Auswahl des Empfängers durch Privilegien und Kontext Der zweite Vorteil, der durch Nutzung der Zertifikate entsteht, ist, daß der Sender den Empfänger nicht nur durch dessen Namen, sondern auch durch dessen Privilegien und den Empfangskontext wie z.B. erlaubte Entschlüsselungszeit festlegen kann.

1. **Senden der Nachricht:** Der Sender O erzeugt die Nachricht MSG, verschlüsselt sie mit einem zufällig erzeugten Schlüssel k und sendet die verschlüsselte Nachricht zusammen mit den Schlüsselinformationen, die k, die Privilegien des Empfängers und dessen Kontext enthalten und mit dem Schlüssel des Senders verschlüsselt sind, sowie das Schlüsselzertifikat von O zum Empfänger R. In Bild 19 werden die Protokollschritte gezeigt.

$$O -> R : \{MSG\}_k, \{k, P, C\}_{K_O}, P, C, KeyCert_{PAS,O}$$

2. **Entschlüsselungsanforderung:** R schickt dem PAS die Schlüsselinformationen, das Schlüsselzertifikat von O, das eigene Privilegzertifikat, das P entspricht und das eigene Schlüsselzertifikat.

$$R -> PAS : \{k, P, C\}_{K_O}, KeyCert_{PAS,O}, PrivCert_{PAS,R,P}, KeyCert_{PAS,R}$$

3. **Entschlüsselungsantwort:** Der Server entschlüsselt die Schlüsselinformationen, und liest k. Nach Überprüfung der Privilegien des Empfängers sowie des Empfangskontext wie z.B Zeitbedingungen schickt PAS den Schlüssel k zurück zum Empfänger R.

$$PAS -> R : \{k\}_{K_R}$$

4. **Entschlüsselung:** Der Empfänger liest k und entschlüsselt so die Nachricht MSG.

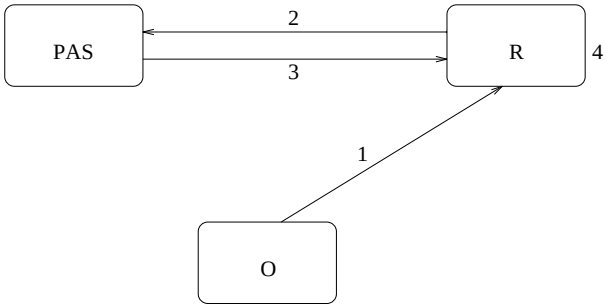

Abbildung 19. Aufrufreihenfolge bei Empfängerauswahl

5.5 Zusammenfassung der Eigenschaften

Die Verwendung von Verfahren mit secret key Zertifikaten eignet sich durch den Einsatz zentraler Server wie dem AS und dem PAS nur für kleinere Netze, hat aber gegenüber Kerberos den Vorteil, daß die Benutzerinformationen nicht zentral gespeichert und geschützt werden müssen. Dennoch muß die Verfügbarkeit der Server sichergestellt werden, was sich wie bei Kerberos durch Replikation erreichen läßt. Public key Zertifikate eignen sich gut um in einem System Rechte und Privilegien fein abgestuft zu definieren und zu verteilen. Ein Schwachpunkt bei secret key Zertifikaten ist, daß sie dem Benutzer der seinen eigenen geheimen Schlüssel kennt, erlaubt eine sog. chosen-plaintext Attacke zu starten. Durch Einführung eines zufälligen Arbeitsschlüssels k, mit dem bei Ausstellung des Schlüsselzertifikats der geheime Schlüssel des Benutzers verschlüsselt wird, kann das verhindert werden. Das Schlüsselzertifikat müsste dann folgendermaßen aussehen:

$$KeyCert_{S,X} = \{X, \{K_X\}_k, \{k\}_{K_S}, L\}^{K_S}$$

6 Bewertung

Der Vergleich der vorgestellten Verfahren zeigt, daß alle ihre Vor- und Nachteile haben. Die folgende Tabelle zeigt Eigenschaften und Unterschiede aufgeschlüsselt nach einzelnen Kriterien.

	Kerberos	X.509	Secret Key Zertifikate
Verschlüsselung	symmetrisch	asymmetrisch, symmetrisch	symmetrisch
Austausch von session keys	ja	ja	ja
Benutzung von Zertifikaten	nein	ja	ja
Speicherung der Benutzerinformationen	zentral, geheim	verteilt, öffentlich	verteilt, geheim
mögliche Systemgröße	mittel groß	sehr groß	klein
Authentisierungsinstanz	authentication server	Benutzer	authentication server
Verwaltung von Privilegien	ja, zentral	nein	verteilt, durch Privileg-zertifikate
Weitergabe von Privilegien	nein	nein	ja, auch mit Einschränkungen
Verbreitung	groß, da z.B. in OSF/DCE eingesetzt	standardisiert	nur Testimple-mentierungen

Die Verwendung von Secret Key Zertifikaten ermöglicht es, Privilegien zu verwalten und weiterzugeben. Durch die Vermeidung der (langsamen) asymmetrischen Verschlüsselung wird jedoch ein zentraler Server nötig, der zur Laufzeit des Systems Zertifikate überprüft.

Die drei vorgestellten Verfahren verhalten sich unterschiedlich bei Änderung des Schlüssels des Servers. Während bei Kerberos sich für den Benutzer sichtbar nichts ändert, werden bei Secret Key Zertifikaten die Anfragen mit veralteten Schlüsselzertifikaten zurückgewiesen. Bei X.509 muß der Rechner des Benutzers prüfen, ob das Zertifikat ungültig geworden ist.

Einsatz objektorientierter Techniken im Netzwerk- und Anwendungsmanagement

Clemens Braun

Kurzfassung

Immer komplexere Netzwerksysteme stellen immer höhere Anforderungen an das Netzwerk- und Anwendungsmanagement. Ein Netzwerkmanagement-System ist laut [Kau95] für Planungsaufgaben, den reibungslosen Netzbetrieb, Fehlersuche und -behebung sowie Konfigurationsverwaltung zuständig und das Anwendungsmanagement regelt die Implementierung, Installation und den Betrieb von Anwendungen. Um diesen Anforderungen gerecht zu werden, müssen die zu verwaltenden Systeme modularisiert werden, um beherrschbar zu bleiben. Dieser Artikel stellt einen Ansatz zur Modularisierung solcher Systeme mit Hilfe objektorientierter Techniken vor.

1 Einleitung

Die Aufgaben des Netzwerkmanagements sind es, für einen möglichst reibungslosen Betrieb des Netzes zu sorgen, Fehlersuche zu betreiben sowie die Planung des Netzes vorzunehmen. Dazu muß das Netzwerkmanagement jede technische oder logische Komponente eines Netzwerkes überwachen und gegebenenfalls deren Zustand ändern. Diese Komponenten können alle Teile sein, die ein Netzwerk ausmachen oder auch nur am Rande mit dem Netzwerk zu tun haben. Die Palette reicht vom einfachen Kabel oder Kanal bis hin zu komplexen Vermittlungssystemen. Ein System, das diese Aufgaben übernimmt, nennt man Netzwerkmanagement-System (NMS). Diese Abkürzung stammt aus [Pyl94]. Das Buch [Sei94] enthält eine gute Einführung zum Thema Netzwerkmanagement, während in [Kau95] das Thema vertieft behandelt wird und einige Probleme aufgezeigt werden.

Netzwerkmanagement-Systeme gibt es in den unterschiedlichsten Ausführungen. Ein Techniker, der anhand von Zustandsanzeigen auf Geräten oder Warnsignalen eine Aktion im Netzwerk vornimmt ist ebenso ein NMS, wie ein Computerprogramm, das auf einer Rechenanlage betrieben wird und Informationen aus dem Netzwerk speichert und anzeigt oder dessen Zustand verändert.

Geht man davon aus, daß ein modernes Telefonnetzwerk aus tausenden verschiedenen Komponenten besteht, aus verschiedenen Technologien und von verschiedenen Herstellern, stellt sich sofort eine Frage: Was macht man, wenn jede Komponente mit ihrem eigenen NMS ausgestattet ist? Die Administratoren solcher komplexen Netzwerke müßten lernen, mit tausenden verschiedenen Systemen umzugehen. Alle NMS könnten verschiedene Schnittstellen aufweisen und während eine Warnung an einem Gerät unbedeutend ist, könnte dieselbe Warnung an einem anderen Gerät zum kompletten Systemabsturz führen. Leider ist dieses Bild von der Realität nicht allzu fern, und noch nicht sehr lange gibt es Bestrebungen, dieses Chaos durch Integration der heterogenen Netze und Netzkomponenten in den Griff zu kriegen.

In Kapitel 2 werden Ansätze beschrieben, durch die auch komplexe Netzwerke übersichtlich und verwaltbar bleiben. Außerdem wird darauf eingegangen, wie verteilte Anwendungen verwaltet werden können. Der Schlüssel dazu ist das Anwendungsmanagement, das sich mit der Implementierung, der Verfügbarmachung und dem Betrieb von Anwendungen beschäftigt. Kapitel 3 beschäftigt sich mit der OO Analyse und dem OO-Entwurf, mit dessen Hilfe man NMSe entwickeln kann. In Kapitel 4 werden dann die objektorientierten Techniken vorgestellt.

2 Integration von Netzwerkmanagement und verteilten Anwendungen

2.1 Entwicklung von Netzwerkmanagement-Systemen

Lange Zeit war das Netzwerkmanagement lokal bei den physischen Bestandteilen des Netzwerkes angesiedelt. Das heißt die Komponenten wie Router, Schalter oder Multiplexer wurden vom Personal direkt überwacht oder geschaltet. Netzwerkmanagementaufgaben werden bei dieser Strategie vor allem dann nötig, wenn Fehler auftreten, die der Betreiber entweder selbst bemerkt oder auf die er vom Benutzer aufmerksam gemacht wird. Vor allem daran ist zu erkennen, daß diese Strategie nicht die benutzerfreundlichste und effektivste sein kann. Solange ein Netzwerk aber vor allem aus elektromechanischen Elementen bestand, gab es wohl keine bessere Alternative.

Im Zeitalter der Computertechnik aber, mit immer leistungsfähigeren elektronischen Bausteinen, vor allem Mikroprozessoren, entwickelten sich auch ganz neue Möglichkeiten für das Netzwerkmanagement. Ein rechnergesteuerten Netzwerk kann nicht nur viel schneller auf auftretende Fehler reagieren, sondern in einem gewissen Maße auch Probleme vorausahnen und durch intelligente Planung vermeiden. Außerdem kann der Personalaufwand gering gehalten werden, was gleichbedeutend mit Wirtschaftlichkeit ist. Das Netzwerkmanagement ist dazu getrennt von den technischen Bausteinen und verwaltet per Kommunikation nicht direkt die physischen Komponenten sondern deren logische Repräsentationen. Durch Ferndiagnose, Fernschalten und Fernparametrieren soll der personalintensive Service-Aufwand vor Ort minimal bleiben. Die Informationen von den Netzwerkelementen werden dabei als Overhead zur Informationsübermittlung innerhalb des Netzes oder über ein eigenes Datennetz übertragen. Mit dem starken Wachstum der öffentlichen und privaten Telefon- und Datennetze wurde diese Art des Netzwerkmanagements das wichtigste Werkzeug, die steigende Komplexität dieser Netze im Griff zu behalten.

2.2 Integration des Netzwerkmanagements

Bis vor kurzem entwickelte jeder Hersteller von Netzwerken oder Netzwerkzubehör eigene Kontrollsysteme für sein Produkt. Es gab lange Zeit keine Normen oder Standards, an die sich die Hersteller hätten halten können. Deshalb tendierte das Netzwerkmanagement dazu, technologie-, produkt- und herstellerspezifisch zu werden. Das Ziel eines NMS sollte

es aber sein, viele Netzwerkelemente verschiedener Fabrikate anzeigen und kontrollieren zu können. Ein NMS, das dies imstande ist zu leisten, nennt man integriertes NMS (INMS).

Von Standardisierungsgremien wurde ein Modell für die Integration heterogener Systeme entwickelt. Dieses Modell nennt sich Systems Management Model und ist in Abbildung 20 dargestellt, die aus [Pyl94] stammt. Eine Realisierung dieses Modells muß sehr umfangreich und komplex sein, um eine Plattform für die Integration von Netzwerkelementen sein zu können. Denn um eine solche Plattform zu entwickeln, muß das Modell sehr fundamental und umfassend umgesetzt werden.

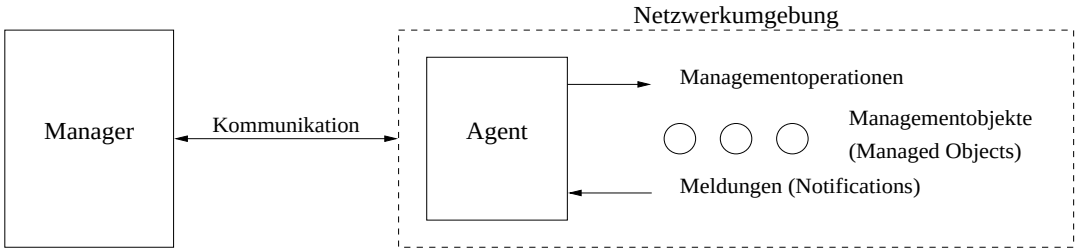

Abbildung 20. Systems Management Model

2.3 Standards im Netzwerkmanagement

Es existieren zwei Standards, die das Modell verwirklichen. Zum einen das Simple Network Management Protocol (SNMP) mir der zugehörigen Datenbasis (management information base MIB), das von der Internet Engineering Task Force (IETF) entwickelt wurde und auf dem Internet mit TCP/IP eingesetzt wird. Dieses Protokoll entstand aus der Aufgabe, das Internet als Weitverkehrsnetz zum Verbinden von lokalen Rechnernetzen handhabbar zu gestalten. Beim Management von TCP/IP-Netzen werden die Managementfunktionen über das SNMP zwischen Managementstation und Netzelement übertragen. Das Netzelement muß, um diese Informationen verarbeiten zu können, über einen Agenten verfügen. Dazu wird jedes Netzelement als eine Menge von abstrakten Datenobjekten modelliert, wie sie in Kapitel 3.2 vorgestellt werden. Das Protokoll wurde ständig weiterentwickelt, und ist daher heute leistungsfähig genug, auf breiter Basis eingesetzt zu werden.

Die International Organization for Standardization (ISO) erarbeitete zusammen mit der International Telecommunications Union - Telecommunications (ITU-T), der früheren CCITT das Common Management Information Protocol (CMIP) und eine dazugehörige MIB zur Verwaltung von OSI-Netzwerken. Dieses Protokoll ist sehr genau spezifiziert und die Konzepte und Begriffe des OSI-Netzmanagements sind in einem Regelwerk festgelegt. Beide kurz vorgestellten Systeme machen mehr oder weniger Gebrauch von objektorientierten Techniken, wie sie in den folgenden Kapiteln vorgestellt werden.

2.4 Anforderungen an das Anwendungsmanagement

Ebenso wie die Netzwerkkomponenten, müssen auch die Anwendungen, die auf dem Netzwerk verteilt ausgeführt werden sollen, verwaltet werden. Das Anwendungsmanagement sollte sowohl solche Anwendungen, die von Anfang an als verteilte Anwendungen für ein Netzwerk entwickelt wurden, als auch Einzelanwendungen, die auf dezentralen Netzwerken installiert werden sollen, verwalten können. Dazu müssen die Anwendungskomponenten um eine Schnittstelle erweitert werden, um vom Anwendungsmanagement kontrolliert werden zu können. Dieser Prozess, der Instrumentierung genannt wird, muß ebenso wie das Anwendungsmanagement selbst, folgenden Anforderungen genügen:

- Allgemeingültigkeit
  Alle Anwendungstypen müssen auf die gleiche Art und Weise in das Managementsystem eingebunden werden können.

- Erweiterbarkeit
  Neue Objekte müssen ohne großen Aufwand in das System integriert werden können. Ohne großen Aufwand heißt, das System soll nicht neu konfiguriert werden müssen und die Modifikation des zu integrierenden Objekts soll nicht zu aufwendig sein.

- Flexibilität
  Das System soll flexibel sein bezüglich dem Verhalten und der Anordnung der verteilten Anwendungen. Das heißt jedes mögliche Verhalten der Anwendung sollte vom Managementsystem verarbeitet werden können.

Ein Management-System, das diesen Anforderungen gerecht werden muß, läßt sich am besten mit Hilfe der objektorientierten Techniken implementieren, die in den Kapiteln 3 und 4 vorgestellt werden. Eine Arbeitsgruppe vom IBM Zurich Research Laboratory erarbeitete einen Ansatz, durch den jegliche Anwendungen durch objektorientierte Techniken so erweitert werden können, daß sie den geforderten Anforderungen genügen [STK94]. Für diese Aufgabe soll eine hierarchische Klassenbibliothek erstellt werden, mit dessen Hilfe der Quellcode der Anwendungen instrumentiert, das heißt mit einer geeigneten Objektschnittstelle (Kapitel 3.3) ausgestattet wird.

3 Objekte

Objekte stellen in ihrer Gesamtheit eine Miniwelt dar, die Abbild der Realität sein soll. Alle für das System wichtigen Elemente werden bei der OO Analyse als Objekte abgebildet. Abhängig von der Anwendung können alle möglichen Gegenstände solche Objekte sein: Eine Person, ein Apfel, ein Schalter, ein Modem, ein Kabel, usw. All diese Objekte besitzen Attribute, z. B.: die Person heißt Herr Müller, das Kabel ist unterbrochen, usw. Desweiteren besitzen die Objekte Funktionalitäten, die Methoden genannt werden, durch die ihre Eigenschaften verändert und beschrieben werden. Z. B. kann die Person mit einer anderen Person reden, ein Modem kann Daten übertragen und der Apfel kann gegessen werden, womit er als Objekt aus dem System verschwindet. Objekte können also auch erzeugt und wieder gelöscht werden. Die Methoden eines Objektes werden entweder als Funktionen oder als Prozeduren implementiert.

3.1 Klassen

Um ein hinreichendes Abbild der Realität zu erhalten, überlegt man sich, aus welchen Komponenten das zu entwerfende System besteht. All diese Komponenten werden dann als Klassen implementiert, aus denen dann beim Betrieb des Systems die Objekte erzeugt werden können. Zuvor muß man sich allerdings noch überlegen, welche Oberklassen, Unterklassen und Instanzen es gibt, man muß eine Klassenhierarchie erstellen. Eine solche ist beispielhaft in Abbildung 21 dargestellt. Es handelt sich dabei um die Vereinbarung von *Scanner*-Objekten nach der *Summarization Function* [Sei94], [10192]. Dabei ist Top die höchste Oberklasse, Unterklassen von Top sind Scanner und Dynamic Scanner. Als Beispiel für die Bedeutungen der Klassen soll der Zusammenhang zwischen Scanner, Heterogeneous Scanner, Homogeneous Scanner und Mean Scanner erläutert werden:

- Die *Managed Object* Class (siehe 3.2) Scanner ist eine nicht instanziierbare Superklasse, die Hilfsmittel zur periodischen Abtastung bestimmter Attribute in spezifizierten Objekten definiert.
- Die Managed Object Class Homogeneous Scanner ist ebenfalls nicht instantiierbar und verfeinert die von der Scanner-Klasse ererbten Eigenschaften um die Möglichkeit zur Selektion abzutastender Attribute sowie dazugehöriger Objekte, wobei die Attribute global zu allen Objekten definiert werden, also in allen Objekten verfügbar sein müssen.
- Daraus wird die instantiierbare Klasse Mean Scanner abgeleitet, welche die Funktionalität aufweist, zu einer Anzahl von Stichproben je Attribut das dazugehörige arithmetische Mittel zu bestimmen.
- Parallel zu der Klasse der Homogeneous Scanner existiert die Managed Object Class Heterogeneous Scanner, die im Gegensatz dazu bereits instantiierbar ist. Mit ihr können unterschiedliche Managed Objects mit verschiedenen Attributen überwacht werden.

Beim OO Entwurf eines NMS werden durch die Objekte eine Möglichkeit bereitgestellt, die physischen oder logischen Komponenten auf verwaltbare Daten abzubilden, mit denen das NMS arbeiten kann. Physische Komponenten sind reale Bauteile und Systeme, wie Schalter, Multiplexer oder Router, die das NMS ebenso benötigt, wie die logischen Komponenten eines Netzes, was Verbindungen, Pfade oder Systemsoftware sein können.

3.2 Managementobjekte

Alle für das Netzwerkmanagement benötigten physischen und logischen Komponenten werden beim Entwurfsprozess zu Objekten, den sogenannten Managementobjekten (Managed Objects). All diese Objekte müssen in einen Regelkreis (Abbildung 22) eingebunden werden können. In diesem Kreis greift das NMS durch Managementoperationen ins Netzwerk ein. Das Netzwerk wiederum zeigt dem NMS seinen Zustand durch Senden seiner Zustandsdaten an. Um an den Managementobjekten diese Funktionalität bereitzustellen, werden die eigentlichen Objekte um eine Schnittstelle erweitert (Abbildung 23),

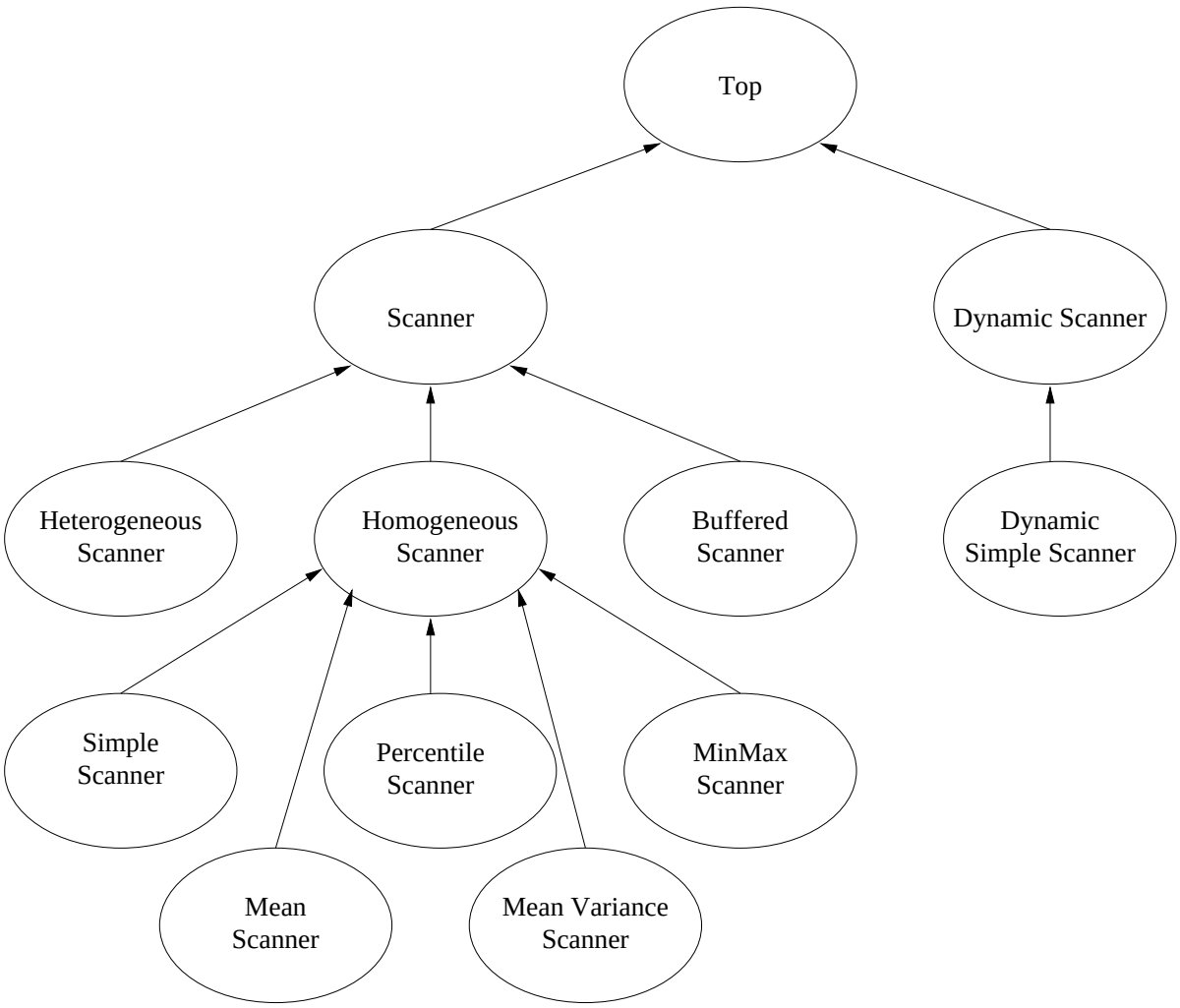

Abbildung 21. Beispiel für eine Klassenhierarchie

die den Datenaustausch mit dem NMS regelt. Zur Erzeugung muß der Benutzer des Managementobjektes dessen Namen und die statischen Managementinformationen wie Typ und Leistungsfähigkeit in die MIB (Management-Informations-Basis) eintragen.

3.3 Objektschnittstellen

Ein Objekt enthält zunächst einen Teil, der dem Geheimnisprinzip (siehe Kapitel 4.2) genügt und vor allem aus internen Datenstrukturen und Algorithmen besteht. Desweiteren enthält es wohldefinierte Schnittstellen, über die es mit anderen Objekten kooperieren und kommunizieren kann. In einem NMS müssen alle Managementobjekte eine Managementschnittstelle aufweisen (Abbildung 23) über die sie mit dem NMS verbunden sind. Diese Schnittstelle besteht aus Attributen des Objektes, Zuständen des Objektes und Managementoperationen. Ein Attribut hat einen Namen und einen Wert. Während der Zeitdauer der Existenz eines Objektes dürfen nur die Werte der Attribute geändert werden, es dürfen jedoch keine Attribute hinzukommen oder gelöscht werden. Der Status beschreibt den momentanen Zustand eines Objektes bezüglich Verfügbarkeit und Operabilität. Die Managementoperationen sind zum Teil Methoden, um die Zustände und Attribute des Objektes verändern zu können. Zum Teil sind es aber auch Methoden, die

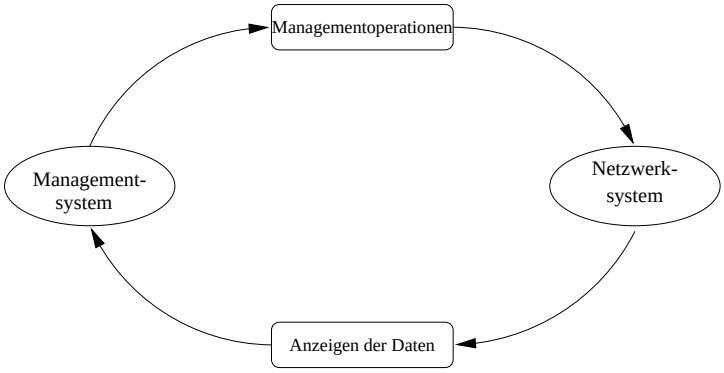

Abbildung 22. Der Management-Prozeß

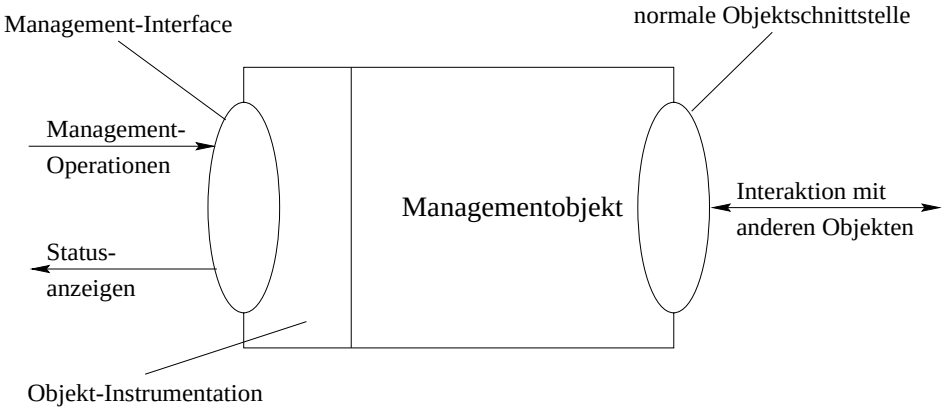

Abbildung 23. Struktur eines Managementobjektes

Meldungen Objektes (sogenannte Notifikationen) über das Netz an das Managementsystem senden können.

4 Objektorientierte Techniken

Zur Integration von NM-Systemen eignen sich hervorragend die objektorientierten Techniken. Diese Techniken können in sowohl in den objektorientierten Sprachen wie z. B. C++, Smalltalk, Eiffel oder Sather angewendet werden, wie auch in anderen nicht objektorientierten Sprachen wie z. B. Pascal. Die OO-Programmiersprachen bieten dabei natürlich bessere Möglichkeiten den objektorientierten Entwurf zu implementieren als die nicht-objektorientierten Sprachen. Die OO-Techniken werden sowohl in [Pyl94], als auch in [Mey90] ausführlich behandelt.

4.1 Generizität

Generische Klassen haben Typen als Parameter, wodurch sie nicht direkt instanziierbar sind, sondern Muster für Unterklassen darstellen. Sie werden auch abstrakte Klassen genannt, wogegen ihre instanziierbaren Unterklassen konkrete Klassen sind. Ein einfaches Beispiel ist in Abbildung 24 dargestellt. Listenelemente können verschiedene Datentypen sein, unter anderem Integer- und Realzahlen. Generizität ist eine Möglichkeit, Hierarchien wie in Abbildung 21 relativ einfach zu verwirklichen.

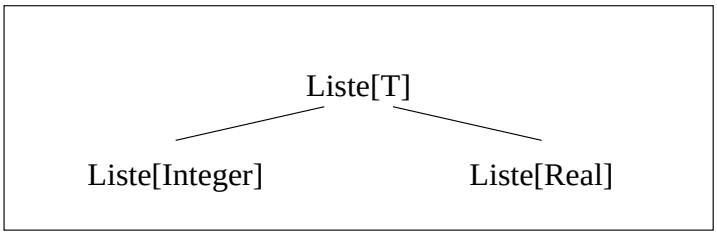

Abbildung 24. Einsatz generischer Parameter

Generizität kann aber auch dazu verwendet werden, verschiedene Sichten eines Objektes verschiedenen Anwendergruppen zugänglich zu machen. Generische Klassen werden dazu mit verschiedenen Schnittstellen ausgestattet. Nur die Elemente der Schnittstelle eines Objektes werden dem Benutzer zugänglich gemacht, während alle anderen Informationen des Objektes geheim bleiben. Beim Netzwerkmanagement ist es dadurch möglich, dem INMS eine große komplexe Schnittstelle einer Komponente zu bieten, während der Benutzer eine sehr einfache Schnittstelle zur Verfügung haben sollte, die auf seine Bedürfnisse zugeschnitten ist und keine Attribute oder Funktionen beinhaltet, die er nicht benötigt.

Dieses Konzept, öffentliche Objektschnittstellen zu gestalten, die eine einfache Sicht auf unter Umständen sehr komplexe Objekte zu gestatten, ist ein zentrales Prinzip im OO-Entwurf. Zugleich ist diese Eigenschaft der Objeke, für verschiedene Leute, verschieden auszusehen, ein erster Schritt ein INMS zu ermöglichen.

4.2 Geheimnisprinzip

Schon mehrmals erwähnt, vor allem im letzten Abschnitt, wurde das Geheimnisprinzip. Es besteht daraus, komplexe und interne Details wie Algorithmen oder Zwischenwerte zu verbergen. Jedes Objekt besteht aus zwei Teilen, der Schnittstelle (public) und den Details der Implementierung (private). Der Benutzer hat nur Zugriff auf den Teil der Attribute und Methoden, die er wirklich braucht und die durch den Ausdruck public (C++-Notation) gekennzeichnet sind. Die versteckten Daten werden in C++ durch private gekennzeichnet.

Gerade die Integration von NM-Systemen ist von sich aus schon sehr komplex, und ein Netzwerk besteht zu weiten Teilen auch wieder aus sehr komplexen Einzelteilen. Desweiteren gibt es die große Zahl verschiedener Modelle eines Elements von verschiedenen Anbietern und Herstellern. Ohne das Geheimnisprinzip wäre es praktisch unmöglich den Überblick über ein INMS zu behalten, zumal sich die Netzwerkarchitekturen, Technologien und Funktionen dauernd weiterentwickeln und ändern.

Außerdem ist es durch das Geheimnisprinzip möglich, Module zu spezifizieren, die sowohl von anderen Modulen, als auch von der Implementierung, unabhängig sind. Solche Module tragen einen großen Teil dazu bei, das Entwurfsprinzip der Wiederverwendbarkeit zu fördern, und sie erleichtern es, Veränderungen im System vorzunehmen. Diese Eigenschaften sind grundlegend für die Integration unterschiedlicher Systeme. Die öffentlichen Attribute und Funktionen der einzelnen Objekte werden in der Systembeschreibung definiert, während der interne Aufbau der Objekte beliebig implementiert werden kann. Erst

diese fest definierten Schnittstellen und Datenstrukturen machen eine Kommunikation, wie sie beim Systems Management Model (Abbildung 20) benötigt wird, möglich.

4.3 Vererbung

Es gibt zwei Wege, eine Klasse zu benutzen. Der einfachere Weg ist der, daß ein Objekt ein oder mehrere Objekte aus anderen Klassen beinhalten darf. In der Literatur wird dieses Verhältnis als Kunde und Lieferant bezeichnet (so z. B. in [Mey90]). Wenn eine Klasse A ein Attribut vom Typ B enthält, heißt A Kunde von B, und B wird Lieferant von A genannt. Es ist sogar möglich, daß eine Klasse Kunde von sich selbst ist, wie das folgende einfache Beispiel zeigt.

```
class BANKKUNDE feature
    name: STRING;
    bürge: BANKKUNDE;
end
```

Der andere Weg, eine Klasse zu benutzen, ist die Vererbung. Durch das Prinzip der Vererbung ist es möglich, Hierarchien wie in Abbildung 21 zu verwirklichen. Die erbende Klasse ist dabei immer eine Spezialisierung seiner Oberklasse. Um ein Beispiel aus Abbildung 21 zu benutzen: Mean Scanner ist Unterklasse von Homogeneous Scanner und erbt damit von der Klasse Homogeneous Scanner. Die Pfeile können bei einer solchen Hierarchiedarstellung als *erbt von* gelesen werden. Ein Mean Scanner ist damit ein spezieller Homogeneous Scanner, der die Eigenschaften aller Homogeneous Scanner wie die Möglichkeit zur Selektion abzutastender globaler Attribute besitzt und desweiteren mit zusätzlicher Funktionalität wie der Bestimmung eines arithmetischen Mittels ausgestattet ist. Diese Sichtweise der Spezialisierung entspricht dem Prinzip des Top-Down-Entwurfs.

Im oben genannten Beispiel hat jede Klasse genau ein Elternteil, es ist aber auch möglich, daß eine Klasse von mehreren Klassen erbt, dieser Prozess wird *Mehrfachvererbung* genannt. Die Mehrfachvererbung ist keine reine Spezialisierung einer Oberklasse, sondern verbindet die Eigenschaften aller Elternteile.

Unterklassen können instanziierbar (konkrete Klassen) oder nicht intanziierbar (abstrakte Klassen) sein. Instanziierbar heißt, aus diesen Klassen können Objekte erzeugt werden. Abstrakten Klassen können dagegen keine Objekte als Instanzen erzeugen, sondern sie werden zur Hierarchiebildung benötigt. Das mag nach unnötigem Mehraufwand klingen, bringt aber in Wirklichkeit eine Vereinfachung mit sich. Durch Blockbildung von ähnlichen Klassen zu abstrakten Oberklassen, wird das System zum ersten übersichtlicher, zum zweiten kann darauf verzichtet werden, den gleichen Programmcode in ähnlichen Klassen zu vervielfältigen.

Die Vererbung ist ebenso wie die Generizität und das Geheimnisprinzip ein wichtiger Schritt auf dem Weg zum Ziel der Erreichung von Wiederverwendbarkeit. Das Maß der Wiederverwendbarkeit wird sehr hoch, wenn die Hierarchie und alle Objektschnittstellen

eines Modelles gut geplant und sauber implementiert worden sind. Genau diese Wiederverwendbarkeit ist es, die das Systems Management Model trotz schnell wechselnder Technologien realisierbar und wirtschaftlich machen kann.

4.4 Allomorphismus

Allomorphismus bedeutet, daß ein Objekt oder eine Funktion in mehreren Klassen definiert sein kann. Abbildung 25 zeigt ein Beispiel, in dem die Klasse Modem zwei Unterklassen hat: Data-Modem und Fax-Modem. Der Ausschnitt des Eiffel-Programmes zeigt, daß die Funktion send-data mehrfach definiert ist: In der Klasse Data-Modem für ein ASCII-Datenobjekt und in der Klasse Fax-Modem für ein Bitmap-Objekt. Ein Aufruf der Funktion in der Form `Modem.send-data (X)` wird diejenige Funktion ausgeführt, die mit dem Parameter X konform ist.

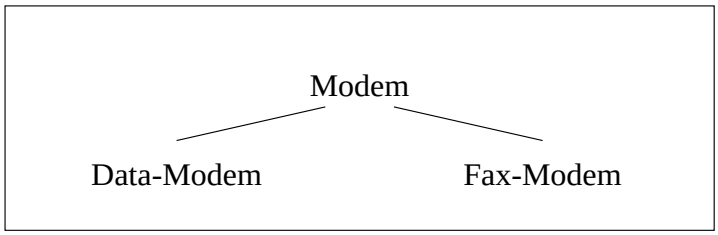

Abbildung 25. Vererbungshierarchie zum Beispiel für Allomorphismus

```
class Modem
feature
     -- allgemeine Attribute und Funktionen
     -- jedes Modemtyps
end; -- class Modem

class Data-Modem
inherit Modem -- erbt alle Attribute und
-- Methoden der Klasse Modem
feature
     .
     .
     send-data (ascii-data-object doc) is
     do
          -- Hier steht die Implementierung der
          -- Funktion fuer Datenmodems.
     end; -- send-data
     .
     .
end; -- class Data-Modem
```

```
class Fax-Modem
inherit Modem -- erbt alle Attribute und
-- Methoden der Klasse Modem
feature
     .
     .
     send-data (bit-map-object doc) is
     do
          -- Hier steht die Implementierung der
          -- Funktion fuer Faxmodems.
     end; -- send-data
     .
     .
end; -- class Fax-Modem
```

Dieses Verhalten hat zwei große Vorteile, vor allem im Hinblick auf die Probleme bei INMSen:

1. Der Hersteller kann einfach neue Unterklassen von Modem entwerfen und dem System hinzufügen, die send-data für andere Datenobjekte enthalten, ohne irgendwo im alten Programmcode etwas verändern zu müssen.
2. Ältere Systeme können mit einem erweiterten Modem-Objekt genauso kommunizieren wie mit der einfacheren alten Version.

Mit dem Prinzip des Allomorphismus ist es möglich, die Komplexität, die durch nebeneinander bestehende Versionen mit unterschiedlichem Verhalten entsteht, in den Griff zu kriegen. Gerade im Bereich des Netzwerkmanagements, das mit den vielen unterschiedlichen Versionen von Netzwerkelementen zurecht kommen muß, ist dieses Prinzip fast unverzichtbar.

5 Zusammenfassung

Für große heterogene Netze ist das Netzwerkmanagement unverzichtbar geworden, und auch für kleine Netze bringt es große Vorteile. Vor allem vom wirtschaftlichen Standpunkt her ist es nötig, mehr in die Planung und Beschaffung gut organisierter Managementsysteme zu investieren. Denn der Betrieb von schlecht organisierten Netzen ist sehr personalintensiv und damit teuer. Es stellt sich aber die Frage, welches NMS eingesetzt werden soll. Die zwei konkurierenden Standards wurden im Kapitel 2.3 vorgestellt. Hier möchte ich nun versuchen, die beiden Protokolle zu vergleichen und sie auf objektorientierte Ansätze zu untersuchen. Einen ausführlicheren Vergleich der beiden Protokoole findet man in [dIF95].

Am häufigsten kommt das SNMP (Simple Network Management Protocol) zum Einsatz. Die Informationseinheiten aus dem Managementinformations-Datenbasis (Management

Information Base, MIB) werden zwar Objekte genannt, von den objektorientierten Techniken wird jedoch nur spärlich Gebrauch gemacht. Die Objekte sind nur einfache Datenstrukturen, oft werden sie deshalb auch treffender als Variablen bezeichnet. Es wird kein Gebrauch vom Geheimnisprinzip gemacht und die Klassen sind nicht wiederverwendbar, da keine Vererbung realisiert ist. Den Erfolg dieses Modells verdankt es wohl überwiegend seiner Einfachheit. Denn ohne langwierige Standardisierungsbemühungen und durch die Beschränkung auf das Wesentliche ließ sich dieses Modell eben wesentlich schneller verwirklichen wie das CMIP und wurde und wird noch immer ständig weiterentwickelt.

Die Grundlage des Common Management Information Protocol (CMIP) der OSI ist auch eine MIB, die darin gespeicherten Informationseinheiten werden aber zurecht als Objekte bezeichnet. Sowohl Attribute, als auch Aktionen und Bedingungen können nach dem Geheimnisprinzip gekapselt werden. Die Klassen sind wiederverwendbar und auch sogar mehrfache Vererbung ist möglich. Es wird also umfassend von den objektorientierten Techniken Gebrauch gemacht. Leider wird das CMIP heute noch ebenso selten eingesetzt wie der gesamte OSI-Protokollturm. Denn wer heute nach einer kleinen wirtschaftlichen Lösung für ein Netzwerk sucht, wird im Rahmen von TCP/IP eher fündig werden als das im ISO/OSI-Umfeld der Fall sein dürfte. Das mag sich in den nächsten fünf Jahren ändern, aber diese Hoffnung hatte man auch schon vor fünf Jahren.

Man darf gespannt sein, was aus der Idee der Klassenbibliothek für die Instrumentation von Anwendungen aus [STK94] wird. Die Realisierung dieser Idee bringt sicherlich einige große Probleme mit sich, denn von den hohen Anforderungen aus Abschnitt 2.4 darf man keine vernachlässigen, wenn das System funktionieren soll. Ein weiteres Problem dieser Idee ist es, daß der Quellcode der Anwendungen vorliegen muß, der aus Gründen des Copyrights von den Herstellern und Verkäufern normalerweise nicht mitgeliefert wird. Ohne Zweifel wäre ein solches System natürlich sehr nützlich. Es ließe sich, sobald es erst einmal vorhanden wäre, auch noch nützlich erweitern. Die Bibliothek könnte mit Hilfe der objektorientierten Techniken problemlos ausgebaut werden, so daß die Anwendungen durch Dienste der Authentifizierung, des Fehlermanagements oder des Accounting erweitert werden. Letzendlich ist es durch diese Vorgehensweise auch möglich, ein System zu entwickeln, das mit anderen Managementsystemen interagieren könnte. Damit hätte man das Ziel der Integration nicht nur von Netzelementen sondern auch von verschiedenen Netzwerkmanagementsystemen erreicht.

Das Adaptionsschichtprotokoll SSCOP

Holger Kraemer

Kurzfassung

Der Artikel beschäftigt sich mit dem Service Specific Connection Orientated Protocol (SSCOP). Dieses Protokoll ist eine Realisierung einer Teilschicht der ATM Adaptionsschicht. Zunächst wird kurz auf ATM eingegangen, dann wird gezeigt wie ATM geschichtet ist und in welcher Teilschicht SSCOP angesiedelt ist. Daran anschließend wird SSCOP in seiner Funktionalität erklärt und anhand eines einfachen mathematischen Modells gezeigt, wie man die wichtigsten Parameter in diesem Protokoll am besten wählt.

1 Einleitung

ATM (Asynchronous Transfer Mode) ist ein paketorientierter Dienst mit dem Vorteil hoher Skalierbarkeit. Mit ATM lassen sich Übertragungsraten von 155 MBit/sec realisieren. Für Breitbandanwendungen hat sich gezeigt, daß konventionelle Protokolle (X.25, LAPB, LAPD...) an ihre Grenzen stoßen. Ein weiteres Beispiel, bei dem diese Protokolle sich als ungeeignet erwiesen haben, war die Satellitenübertragung, weil hier der sogenannte round trip delay enorm hoch ist. Unter round trip delay ist die Zeit zu verstehen, die das Signal vom Sender zum Empfänger und vom Empfänger zurück zum Sender unterwegs ist. Als Konsequenz daraus wurde für ATM ein eigener Protokollstack entwickelt, der diese speziellen Anforderungen besser bewältigen kann. ATM wurde als dreischichtiges Protokoll entworfen. Es besteht aus einer physikalischen Schicht, einer ATM Schicht und der ATM Adaptionsschicht, wie in Abbildung 1 dargestellt.

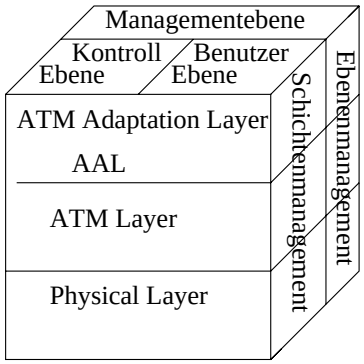

Abbildung 26. ATM Schichtenmodell

In dieser Ausarbeitung geht es darum, SSCOP (Service Specific Connection Oriented Protocol) vorzustellen. SSCOP ist in die ATM-Adaptionsschicht (AAL) einzuordnen und beinhaltet viele Protokollmechanismen, die für ein High Performance Protokoll notwendig sind. Die AAL gliedert sich in eine Convergence Sublayer (CS) sowie eine Segmentation and Reassembly Sublayer (SAR), wie aus Abbildung 2 zu entnehmen ist.

Die Convergence Sublayer ihrerseits besteht aus den beiden Teilschichten Service Specific Convergence Sublayer (SSCS) und Common Part Convergence Sublayer (CPCS), wie man ebenfallsAbbildung 2 entnehmen kann. Das Protokoll SSCOP, das in dem folgenden Artikel behandelt wird, ist eine Variante der SSCS.

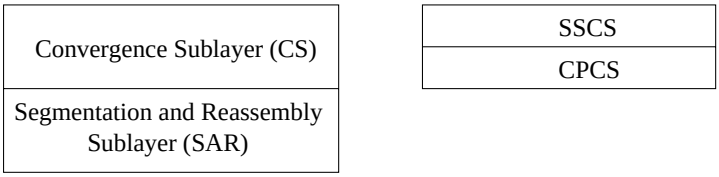

Abbildung 27. Schichtung der ATM Adaptionsschicht

2 ATM und das ATM Schichtmodell

In diesem Abschnitt sollen zunächst einmal die Eigenheiten von ATM dargestellt werden. Anschließend soll versucht werden, das ATM Schichtmodell mit dem OSI Protokollturm zu vergleichen. ATM ist ein paketorientierter Dienst, d.h. die Zuteilung der Bandbreite ist nicht statisch festgelegt, sondern erfolgt variabel. Durch die virtuellen Pfade und virtuellen Kanäle ist ATM verbindungsorientiert. Die ATM Pakete werden als Zellen bezeichnet und sind 53 Byte lang, wobei 5 Byte für den Header reserviert sind und 48 Byte als Nutzdaten zur Verfügung stehen. Unter asynchron ist zu verstehen, daß die an ATM übergebenen Daten nicht periodisch sein müssen. Auf der Leitung selbst werden die Zellen kontinuierlich gesendet. Innerhalb der physikalischen Schicht werden Leerzellen eingefügt wenn nicht genug Zellen vom Benutzer übergeben wurden.

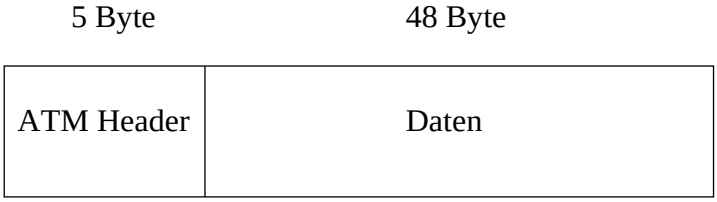

Abbildung 28. ATM Zelle

Der ATM Header dient dazu, die Zellen durch das Netz zu routen. Er enthält weiterhin eine Prüfsumme über den Zellkopf und zeigt an, ob es sich im Datenfeld um Benutzerdaten oder um Netzwerkdaten handelt. Der Header sieht im einzelnen folgendermaßen aus.

Dabei bedeuten die Felder:

VPI (Virtual Path Identifier): Der Virtual Path Identifier ist eine logische Einheit der ATM Adresse. Ein virtueller Pfad kann mehrere VCI (Virtual Channel Identifier) enthalten. Der VPI ist immer nur zwischen zwei Netzwerknoten gültig.

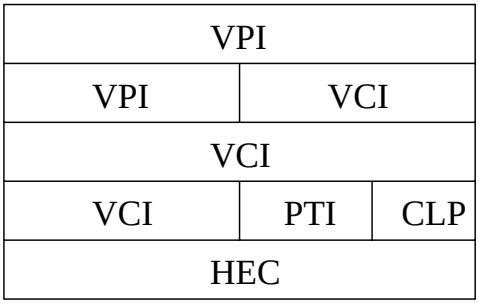

Abbildung 29. ATM Header

VCI (Virtual Channel Identifier): Der Virtual Channel Identifier ist ebenfalls eine logische Einheit der ATM Adresse und gilt auch nur innerhalb eines virtuellen Pfades. Wenn die virtuellen Kanäle innerhalb eines Pfades liegen ändern sich die VCI nicht.

PTI (Payload Type Information): Dieses Feld enthält Informationen über die Daten im Datenfeld und ist 3 Bit lang. Daraus ergeben sich 8 mögliche Kombinationen zur Klassifizierung der Nutzdaten.

1. 000 User Data Cell, congestion not experienced , ATM Layer User to ATM Layer User = 0
2. 001 User Data Cell, congestion not experienced, ATM Layer User to ATM Layer User = 1
3. 010 User Data Cell, congestion experienced, ATM Layer User to ATM Layer User = 0
4. 011 User Data Cell, congestion experienced, ATM Layer User to ATM Layer User = 1
5. 100 Segment OAM F5 flow cell
6. 101 End to End OAM F5 flow cell
7. 110 Future reserve
8. 111 Future reserve

CLP (Cell loss priority): Falls dieses Bit gesetzt ist, wird die Zelle bei Engpässen im Netz eher vernichtet als bei nicht gesetztem Bit.

HEC (Header Error Detection/Correction)

Eine Einordung von ATM in das OSI Schichtenmodell ist nicht ohne weiteres möglich. ATM liegt mit seiner Funktionalität zwischen den Schichten 2 und 4. Die Einordnung in die Schicht 4 ist deswegen möglich, weil ATM Quality of Service Parameter übergeben werden können, und weil es eine gesicherte Ende zu Ende Verbindung über ein Subnetz zur Verfügung stellt. Es werden auch Routing Aufgaben, die der Schicht 3 zuzuordnen sind angeboten.

Die Einordnung von ATM in die OSI Schicht 2 ist deshalb gerechtfertigt, weil ATM teilweise unter TCP/IP genutzt wird und somit nur die Aufgaben der Sicherungsschicht, nämlich einen gesicherten Kanal zur Verfügung zu stellen, übernehmen muß. Wie schon in Abbildung 1 zu sehen war, gibt es für ATM ein eigenes Schichtenmodell. Im Gegensatz zum OSI Modell handelt es sich nun um ein dreidimensionales Gebilde. Neben den bekannten Schichten kommen hier noch die Säulen oder Ebenen hinzu. Im einzelnen handelt es sich hier um die Benutzerebene, die Managemetebenen und die Kontrollebene. Die Benutzerebene dient dazu, die Daten des Benutzers zu übertragen und den Datentransfer zu regulieren. Sie entspricht somit in etwa dem B-Kanal beim Schmalband-ISDN. Die Mangementebene teilt sich noch einmal in das Ebenenmanagement, welches für die Koordination zwischen den einzelnen Ebenen verantwortlich ist, und das Schichtenmanagement, das die Management-Funktionen für die spezifischen Schichten entält. Als dritte Ebene gibt es noch die Kontrollebene, die die Übertragung der Signalisierungsinformationen übernimmt.

3 Funktionalität der ATM Schicht

Die ATM-Schicht stellt der ATM-Adaptionsschicht eine Reihe von Diensten zur Verfügung, die sie durch die Dienste der physikalischen Schicht realisiert. Zu diesen Diensten gehören:

- Cell Multiplexing / Demultiplexing: Diese Funktion stellt beim Sender einen zusammengesetzten Zellstrom her, der Zellen mit verschiedenen VPI bzw. VCI zusammensetzt. Auf der Empfängerseite wird der Zellstrom wieder zerlegt und entsprechend weitergeleitet.
- Cell Relaying: Das Cell Relaying betrifft das Routing, d.h. die Auswertung der VPI bzw. VCI.
- Delay Priority Processing: Die Priorität wird beim Verbindungsaufbau ausgehandelt. Die Funktion nutzt die VPI/VCI, um die entsprechende Verbindung zu ermitteln.
- Cell Loss Priority Processing: Diese Funktion utzt das Feld CLP um entsprechend gekennzeichnete Zellen zu vernichten.
- Connection Assignment: Das Connection Assignment ordnet bestimmte Verbindungen bestimmten ATM Funktionen und Endpunkten zu. Hierbei wird es vom ATM-Management gesteuert.
- Connection Removal: Diese Funktion stellt das Gegenteil von Connection Assignment dar. Auch sie wird durch das ATM-Management gesteuert und beendet Assoziationen innerhalb der ATM Schicht.
- Cell Construction: Dient dazu, die Header Felder zu generieren. Dabei benutzt es Informationen der Funktion Connection Assignment.
- Unassigned Cell Construction: Diese Funktion nimmt den Zellfluß der belegten Zellen und fügt leere Zellen entsprechend der Transferrate der physikalischen Schicht ein. Hierbei wird ein eindeutiges VPI/VCI Feld verwendet.

- Unassigned Cell Extraction: Gegenstück zu Unassigned Cell Construction. Diese Funktion nimmt die nicht belegten Zellen und vernichtet sie.
- Cell Copying: Die Funktion kopiert den Datenteil zur ATM-Mangement Instanz. Sie wird durch ATM Mangement gesteuert.
- Cell Reception: Hier wird da PTI-Feld ausgewertet. Diese Daten werden teilweise an das ATM-Management weitergegeben.

4 ATM Adaptionsschicht

Die ATM Adaptionsschicht stellt dem ATM User Dienste zur Verfügung, die sie unter Nutzung der ATM Schicht erbringt. Man kann die Dienste der ATM Adaptionsschicht in 4 Klassen unterteilen:

- A asynchroner und synchroner Transport (gleichbleibende Bitrate)
- B variable Bitraten
- C verbindungsorientierter Dienst mit variabler Bitrate
- D verbindungsloser Dienst mit variabler Bitrate

Die Dienstklassen sind nicht mit den AAL-Typen zu verwechseln. Momentan gibt es 4 AAL-Typen und für jeden Diensttyp A bis D steht mindestens ein AAL-Typ zur Verfügung. Die AAL-Schicht läßt sich in die Segmentation and Reassembly und die Convergence Sublayer unterteilen. Die SAR nimmt Daten von der CS, segmentiert sie und fügt Header oder Trailer zum zusammensetzen oder zur Fehlererkennung ein. Die fertigen Blöcke, die eine Länge von 48 Byte haben, gibt sie an die ATM-Schicht weiter. Die Convergence Sublayer bietet Funktionen zur Fehlererkennung und zur Flußkontrolle an. Die Header bzw. Trailer dieser Schicht werden um das von oben entgegengenommene Paket gebaut, was bedeutet, daß sie nicht um jede einzelne Zelle gebaut werden. Bis

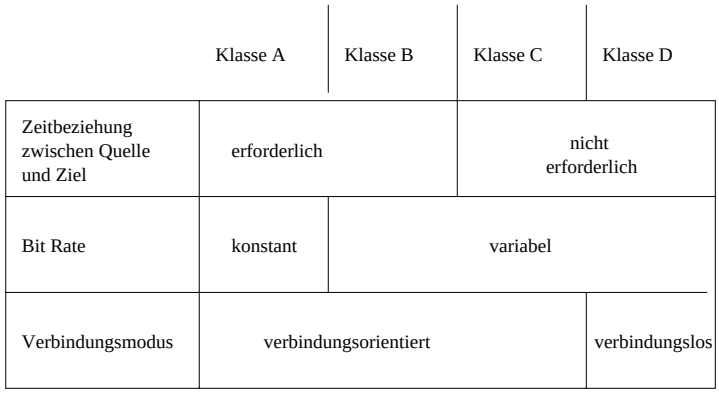

Abbildung 30. ATM Service Klassen

zum jetzigen Zeitpunkt wurden 4 ATM Adaptionlayer Typen spezifiziert, die jetzt im einzelnen näher beschrieben werden sollen.

– AAL Typ 1: Der AAL Typ 1 unterstützt speziell 64 kbps Ströme und stellt die Dienste Zeiterkennung, Zeitanzeige, Synchronisation und die Anzeige von Verlusten zur Verfügung. Dieser Typ ist verbindungsorientiert. Der AAL Typ 1 hat einen ein Byte langen Header, der wie folgt aufgebaut ist. Dabei bedeuten:

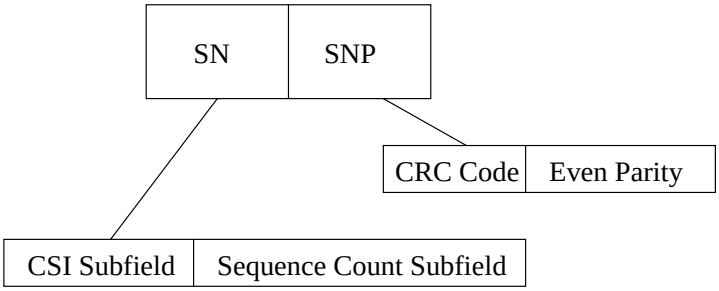

Abbildung 31. Header des AAL Typs 1

- SN: Sequence Number, Länge 4 Bit.
- SNP: Sequence Number Protection field, Länge 4 Bit.
- CSI: Convergence Sublayer Indication, Länge 1 Bit.
- SC: Sequence Count, Länge 3 Bit.
- CRC: Code über SN Feld.
- Parity: gerade Parität über die ersten 7 Bit.

Die Convergence Sublayer des Typ 1 ist noch nicht vollständig spezifiziert.

– AAL Typ 2: Der AAL Typ 2 unterstützt variable Bitraten mit Timing. Momentan sind zu diesem Typ aber weder die SAR noch die Convergence Teilschicht spezifiziert.

– AAL Typ 3/4: Der AAL Typ 3/4 unterstützt eine Ende-zu-Ende-Kommunikation mit variabler Bitrate. Bei diesem AAL-Typ ist die Convergence Sublayer noch einmal unterteilt, und zwar in die Service Specific Convergence Sublayer (SSCS) und zum anderen in die Common Part Convergence Sublayer (CPCS). Dies wurde schon in Abbildung 1 deutlich gemacht. Die CPCS bietet den ungesicherten Transport mit variabler Bitrate. Für den gesicherten Transport wird die Funktionalität der SSCS benötigt. Die SSCS benutzt zur Realisierung eines gesicherten Transportes Sequenz-Nummern und Übertragungswiederholungen. Beim AAL Typ 3/4 sieht eine SAR-PDU wie in Abbildung 7 dargestellt aus.

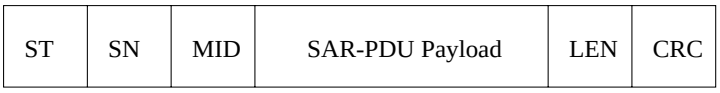

Abbildung 32. Segmentation and Reassembly PDU der AAL 3/4

Dabei bedeuten:

- ST: Sequence Type (BOM, COM, EOM = Begin, Continuation, End Of Message oder SSM = Single Segment Message).
- SN: Sequence Number, modulo 16.
- MID: Der Message Identifier dient dazu, verschiedene AAL-Verbindungen zu einer ATM-Verbindung zu multiplexen.
- LEN: Anzahl der Bytes mit Nutzdaten (0 - 44).
- CRC: Cyclic Redundancy Check.

Die CPCS PDU, also diejenige PDU, die eine Teilschicht höher vorliegt, ist in Abbildung 8 dargestellt.

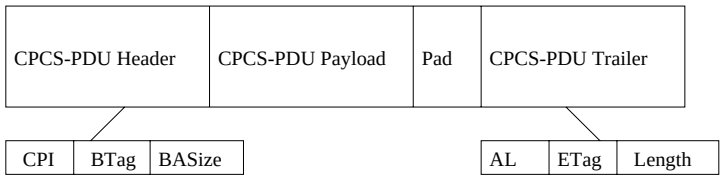

Abbildung 33. CPCS PDU der AAL 3/4

Dabei bedeuten:

- CPI: Common Part Indicator.
- Btag: Fehlerkontrollfeld, Etag im Trailer enthält denselben Wert.
- BA-Size: Entweder Länge der Nutzdaten oder die maximale Länge.
- AL: Füllt den Trailer derart, daß er an die 32 Bit Grenze geht.
- ETag: siehe BTag.
- Length: Länge der Nutzdaten ohne PAD.

– AAL Typ 5: Der AAL Typ 5 unterstützt speziell die Anforderungen der Klasse C, daß heißt einen verbindungsorientierten Dienst mit variabler Paketlänge. Dieser Typ ist eine vereinfachte Version des Typs 3/4 und ist dafür gedacht unter, TCP/IP zu arbeiten. Die wichtigsten Unterschiede zum Typ 3/4 bestehen darin, daß es keine gesicherte Übertragung gibt, da diese ohnehin von z.B. TCP/IP zur Verfügung gestellt wird. Zum anderen entfällt das Multiplexing, da dies bereit von der unter der AAL-Schicht liegenden ATM-Schicht übernommen wird. Die AAL-Typ 5 ist hier wichtig, weil SSCOP über der AAL Typ 5 arbeitet. Eine CPCS PDU des AAL Typs 5 ist in Abbildung 9 dargestellt.

  Dabei bedeuten:

  - UU: Für Benutzer reserviert.

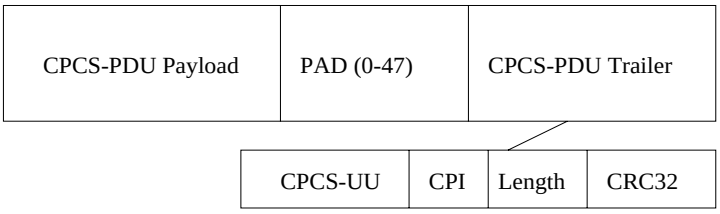

Abbildung 34. CPCS PDU der AAL 5

- CPI: Common Part Identifier.
- Length: Nutzdaten in Byte.
- CRC32: Cyclic Redundancy Check.

Auffallend beim AAL-Typ 5 ist, daß es weder einen Header noch einen Trailer gibt, sondern daß die SAR PDU mit ihrer ganzen Breite von 48 Byte Nutzdaten enthält.

5 SSCOP Überblick

SSCOP (Service Specific Connection Oriented Protocol) ist ein Hochleistungs-Sicherungsprotokoll (GBit/sec), das auch hohe Übertragungsverzögerungen, wie sie beispielsweise bei der Übertragung durch einen Satelliten entstehen, verkraften kann. SSCOP stellt eine X.212 Schnittstelle für die höheren Schichten zur Verfügung. Weiterhin bietet SSCOP eine Ende-zu-Ende-Kommunikation, Mechnismen zur Flußkontrolle und ein selektive Übertragugswiederholung an. Ein Vorteil von SSCOP ist, daß es keinerlei Angaben über den round trip delay benötigt. SSCOP ist in der Lage, die Länge seiner Datenpakete an die Verlustrate anzupassen, die gerade auf dem Netz herrscht. Es wurde speziell für die Signalisierung entworfen und bietet eine ähnliche Funktionalität wie LAPD bei ISDN. Je nach Bedarf werden bestimmte Teile der Funktionalität von SSCOP den verschiedenen Anforderungen zur Verfügung gestellt. Die Anforderungen wiederum sind zu bestimmten Service Specific Coordination Functions zusammengefasst. Die beiden ersten standardtisierten SSCF wurden zwar für die User-To-Network Interface (UNI) bzw. die Network-To-Network Interface (NNI) Signalisierung entworfen, aber eine weitere SSCF, die die Funktionalität der OSI-Schichten 3 und 4 übernehmen soll, wird gerade entwickelt. Es bleibt noch zu bemerken, daß SSCOP nicht auf ATM beschränkt ist, sondern sich auch für andere Umgebungen mit hoher Leitungsverzögerung oder mit stark rauschendem Kanal gut eignet.

6 Beschreibung der Funktionalität von SSCOP

SSCOP bietet eine gesicherte Übertragung von Frames variabler Länge an. Zu diesem Zweck werden die verschiedenen Frames solange beim Sender zwischengespeichert, bis sie vom Empfänger bestätigt wurden.

Um verlorengegangene oder verfälschte Pakete zu reparieren, benutzt SSCOP den Mechanismus der selektieven Übertragungswiederholung, was bedeuted, daß nur die Pakete, die auch tatsächlich verloren gegangen sind, erneut übertragen werden. Ein weiterer in anderen Protokollen verwendeter Mechanismus, der als Alternativeebenfalls denkbar wäre, ist die Go Back N Strategie. Diejenige Prozedur, die die erneute Übertragung verlorengegangener Pakete in Gang setzt, heißt ARQ (automatic repeat request). Die Entscheidung, ob überhaupt Frames verlorengingen, wird mit Hilfe von Sequenznummern getroffen. Der Empfänger bestätigt dem Sender den Empfang der Pakete. Falls der Empfänger ein Paket vermißt, fordert er beim Sender explizit eine erneute Überrtagung genau dieses einen Paketes an. Der Sender wiederum überträgt nur die Pakete erneut, die vom Empfänger explizit noch Mal angefordert wurden, das bedeutet insbesondere, daß es keine Time Outs gibt und somit auch Timer überflüssig werden. Für den Empfänger bedeutet dieses Verfahren, daß er einen Puffer zur Verfügung stellen muß, in dem er die korrekt empfangenen und die ursprünglich verlorengegangenen Frames wieder in die richtige Reihenfolge bringen kann.

Die Flußkontrolle bei SSCOP wird duch eine Schiebefensterprotokoll (sliding window protocol) mit variabler Fenstergröße realisiert. Die Größe des Fensters kann vom Empfänger dynamisch verändert werden.

SSCOP benutzt vier Rahmentypen, wobei einer dieser 4 Frame-Typen für Sequenced Data (SD) und die anderen drei für Flußkontrolle vorgesehen sind. Bei den drei für die Flußkontrolle vorgesehenen Paketen handelt es sich im einzelnen um POLL, STAT und USTAT. SD Frames haben eine variable Länge von bis zu 65636 Byte. Jeder dieser Rahmen kann eine Sequenznummer modulo 2^{24} enthalten.

POLL-Frames werden vom Sender verschickt und veranlassen den Empfänger dazu, seinen aktuellen Status an den Sender zurückzuschicken. Ein POLL-Frame enthält die Sequenznummer des Paketes, das der Sender als nächstes SD-Frame verschickt, sowie eine POLL Sequenz Nummer, die wie eine Art Timer zu verstehen ist. Ein POLL wird auf Senderseite entweder nach Ablauf eines Timers generiert oder alternativ dazu nach dem Absenden einer bestimmten Anzahl von Daten-Rahmen.

STAT-Frames werden vom Empfänger an den Sender geschickt. Sie beinhalten Informationen wie die Fenstergröße, d.h. eine Sequenznummer, die der Sender nicht überschreiten darf, die Sequenznummer des nächsten erwarteten Daten-Frames, die vom Sender empfangenen POLL-Sequenz-Nummer (wird eins zu eins kopiert) und eine Liste der verlorengegangenen Daten-Frames. Die verlorengegangenen Datenpakete erkennt der Empfänger anhand von Lücken in den Sequenznummern, die sich in dem Puffer befinden, sowie der Nummer des nächsten zu verschickenden Paketes (die vom Sender im POLL-Frame eingetragen wurde). Der Sender nutzt die im STAT-Feld enthaltenen Daten, um die vom Empfänger bestätigten SD-Frames aus seinem Puffer zu entfernen, die vom Empfänger als verloren angegebenen SD-Frames noch einmal zu übertragen und die Größe des Übertragungsfensters entsprechend den Angaben des Empfängers einzurichten.

USTAT-Frames (unsolicited status) werden ebenfalls vom Empfänger an den Sender geschickt. Sie werden unmittelbar nach Erkennen eines verlorenen Daten-Frames abgeschickt und fordern beim Sender die erneute Übertragung genau des einen verlorengegan-

genen Datenpaketes an. Der Unterschied liegt zum einen darin, daß das USTAT Frame nicht eine ganze Liste von verlorengegangenen Daten-Frames beinhaltet, sondern nur ein einziges, zum anderen darin, daß das USTAT Frame unabhängig von einem POLL an den Sender verschickt wird. Eine weitere Besonderheit bei einem USTAT-Frame besteht darin, daß es nur einmal pro verlorengegangenem Daten-Frame verschickt wird. Geht dieses Frame erneut verloren, so folgt kein USTAT mehr. Dieser Mechanismus soll noch einmal anhand eines Beispiels verdeutlicht werden. In Abbildung 10 ist zu sehen, daß der Sender die Pakete 0 bis 6 fehlerfrei an den Empfänger Übertragen kann, bei Paket Nr. 7 tritt ein Fehler auf, den der Empfänger aber erst nach Empfang von Paket 8 bemerkt. Daraufhin schickt er ein USTAT an den Sender zurück. Den Sender erreicht dieses USTAT unmittelbar nach dem Versenden seines zweiten Polls, und er überträgt das Paket 7 noch einmal. In dem STAT, das der Sender nun aufgrund seines Polls erhält, ist das Paket 7 noch als fehlend eingetragen, wird aber nicht noch einmal verschickt.

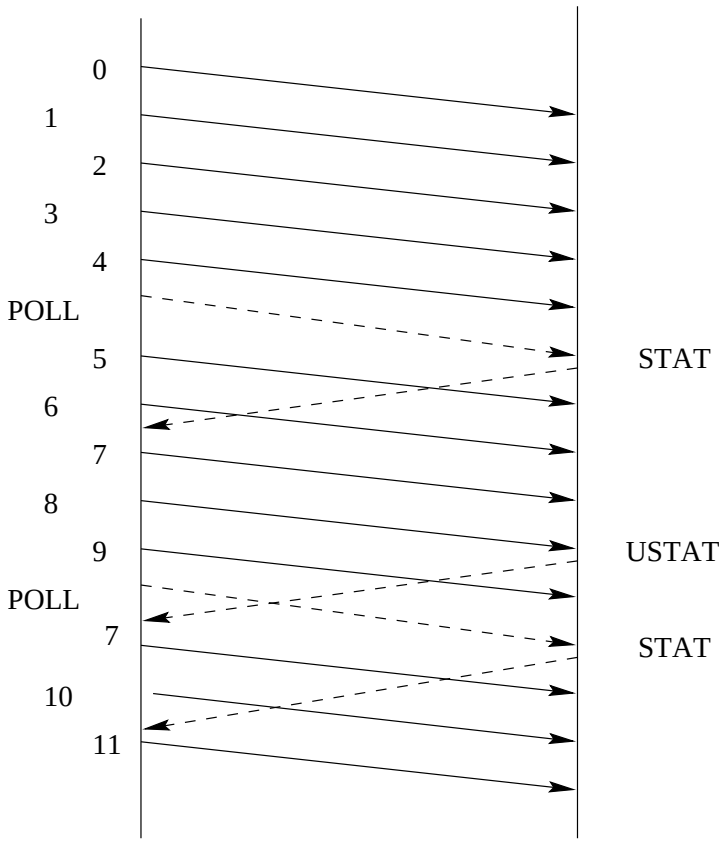

Abbildung 35. Beispiel der Flußkontrolle

7 Analyse der einzelnen Parameter von SSCOP

Bei SSCOP sind die wichtigsten Parameter die Größe des Fensters, das der Empfänger dem Sender für die Pufferung der Daten zur Verfügung stellt, die Größe der Datenrahmen, die SSCOP der darüberliegenden Schicht zur Verfügung stellt, und das Intervall zweier aufeinanderfolgender Polls des Senders, mit dem er die Empfangsbestätigung der

versendeten Daten beim Empfänger anfordert. Ziel ist es, den maximalen Durchsatz durch das Netz zu optimieren und dabei das Fenster möglichst klein, die Paketgröße möglichst groß und das Intervall zwischen zwei Polls möglichst groß zu halten. Die Minimierung der Fenstergröße hat einfach den Vorteil gesparten Speicherplatzes, die Maximierung der Paketlänge birgt den Vorteil, den Overhead innerhalb eines Paketes zu reduzieren und die Ausdehnung des Poll-Intervalls führt dazu, den Protokolloverhead zu verringern. In diesem Abschnitt geht es nun darum, ein mathematisches Modell vorzustellen, mit dem es möglich ist den maximalen Protokolldurchsatz in Abhängigkeit von der Fenstergröße und dem Abstand zwischen zwei Polls darzustellen. Um diese Modell nun näher zu erläutern, sind einige Definitionen nötig.

- r stellt die Bitrate dar.
- rtd steht für den round trip delay, d.h. die Zeit, die ein Paket vom Sender zum Empfänger und wieder zurück an den Sender unterwegs ist.
- Tr gibt die Anzahl der Pakete an, die innerhalb des round trip delay abgesendet werden können. Es gilt die Beziehung: $Tr = r * rtd/(8 * s)$.
- s soll die Rahmengröße angeben.
- e steht für die Bitfehlerrate.
- p steht für die Rahmenfehlerrate, wobei hier vom Zusammenhang $p = 1 - (1 - e)^{8s}$ ausgegangen werden soll. Desweiteren soll $e < 10^{-7}$ gelten.
- Tp gibt die Anzahl der Datenpakete an, die zwischen zwei Timer-Polls verschickt werden. Für Tp gilt also $Tp = r * TimerPoll/8s$.
- W gibt die Größe des Fensters in Rahmen an.

Weiterhin werden einige Anforderungen an das Modell gestellt, um eine gewisse Vereinfachung zu erhalten. Dazu zählen:

- Die Zeitachse hat nur diskrete Werte mit Intervall $T = 8s/r$. Dies entspricht der Rahmenübertragungszeit. Weiter seien TR und TP die nach oben aufgerundeten Werte von Tr und Tp.
- Der Datenfluß wird als unidirektional vorausgesetzt. POLL und STAT messages hingegen werden auf einem bidirektionalen, verlustfreiem Kanal übertragen.
- Alle erneuten Übertragungen von verlorengegangenen Daten verlaufen erfolgreich.
- Für jedes verlorengegangene Frame wird ein USTAT erzeugt, d.h. insbesondere, daß keine zwei aufeinanderfolgenden Frames verloren gehen.
- Dem Sender stehen Sendedaten in ausreichender Menge zur Verfügung.

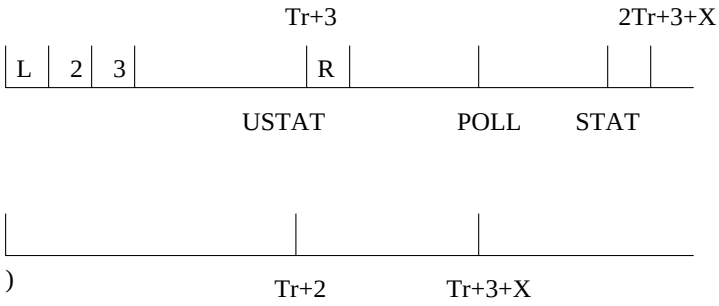

Abbildung 36. Beispiel der Flußkontrolle

Unter der maximalen Durchsatzeffizienz verstehen wir $p_{max} = \frac{erfolgreiche\ Frames}{alle\ Frames}$. Es soll nun versucht werden, die Fenstergröße in Abhängigkeit des round trip delay und des Poll Intervalls zu betrachten und für die einzelnen Beziehungen dieser Parameter den Durchsatz zu bestimmen. Dies soll in Abbildung 11 verdeutlicht werden. Die Abbildung zeigt die Situation aus Sicht des Senders. Falls das erste Paket verlorengeht, was hier durch L angedeuted wird, dauert es $2 + Tr/2$, bis der Empfänger das nächste Paket empfängt und somit den Verlust bemerkt. Er reagiert darauf, indem er sofort ein USTAT an den Sender zurückschickt. Dieses USTAT erreicht den Sender zum Zeitpunkt $Tr + 3$. Dieses Paket muß jetzt noch mit einem STAT bestätigt werden. Ein POLL wird zum Zeitpunkt $TR + 3 + X$ abgeschickt, wobei X im Bereich $0 \leq X \leq TP - 1$ liegt. Das zugehörige STAT erreicht den Sender also zum Zeitpunkt $2 * TR + 3 + X$. Aus diesem Zusammenhang ergeben sich nun 3 wichtige Fälle für die Fenstergröße.

1. $W > 2TR + TP$ In diesem Fall muß der Sender nie auf den Empfänger warten. Im ungünstigsten Fall tritt das POLL erst $TP - 1$ Zeitintervalle nach dem Anfang der erneuten Übertragung auf (also $X = TP - 1$). Das entsprechende STAT trifft zum Zeitpunkt $2 * TR + TP + 2$ ein. Zu diesem Zeitpunkt ist das Fenster des Senders noch nicht übergelaufen, was bedeutet, daß der Sender nie auf den Empfänger warten muß. Hier gilt für den Durchsatz:
$$p_{max} = 1 - p$$

2. $TR+TP \leq W \leq 2TR+TP$ In diesem Fall muß der Sender immer dann warten, wenn Wiederholungen erforderlich geworden sind. Die Fenstergtöße ist so ausgelegt, daß der Puffer nicht überläuft, wenn keine erneuten Übertragungen notwendig werden, denn es besteht ja die Möglichkeit TP Frames, also alle Frames zwischen zwei Polls, und TR Frames, also die Anzahl von Frames die innerhalb eines round trips verschickt werden, zwischenzuspeichern. Hier gilt für den Durchsatz:
$$p_{max} = 1 - p(1 + \frac{1}{TP} \sum_{k=0}^{TP-1} max(0, 2TR + TP - W - k + 1))$$

3. $W < TR + TP$ Jetzt muß der Sender auch dann auf den Empfänger warten, wenn alle Übertragungen korrekt verlaufen sind. In diesem Fall ist das Fenster kleiner als die Anzahl der Frames, die vom abschicken des Polls bis zum Empfange des STAT eignetlich zwischengespeicher werden müßten. Hier kommt eine starke Abhängigkeit

von TP zu TR hinzu und für den Durchsatz gilt:

$$p_{max} = \frac{W}{TP}, \qquad \text{falls } TP \geq TR \text{ und } W < TP - TR$$

$$p_{max} = \frac{W + TP - TR}{2TP}, \qquad \text{falls } TP \geq TR \text{ und } TP - TR < W < TP + TR$$

$$p_{max} = \frac{W}{TR + TP}, \qquad \text{falls } TP < TR$$

8 Zusammenfassung

In dieser Ausarbeitung wurde das Service Specific Connection Oriented Protocol SSCOP vorgestellt. Hierzu wurde kurz auf ATM und das dazugehörende Schichtenmodell eingegangen. Es wurde dann eine Einordnung von SSCOP in dieses Modell vorgenommen und versucht, einen Bezug zum OSI Modell herzustellen. Daran anschließend wurde SSCOP im Detail erklärt, und es wurde aufgezeigt wo die Stärken dieses Hochgeschwindigkeitsprotokolls liegen. Im letzten Teil ging es darum ein Modell zu erstellen, mit dem sich die einzelnen Parameter, die in SSCOP gesetzt werden können mathematisch optimieren ließen. Insbesondere hat sich gezeigt, daß die Fenstergröße einen entscheidenden Einfluß auf den Durchsatz ausübt. Im letzten Kapitel daher noch auf diesen Zusammenhang näher eingegangen.

Adreßauflösung für Multicast-Gruppen mit MARS

Michael Hocke

Kurzfassung

Die Implementation von IP und den dazugehörigen Multicast-Diensten auf ATM-Netzwerken ist eine nicht-triviale Aufgabe, da ATM nur verbindungsorientiert arbeitet. Der Text beschreibt, wie eine solche Abbildung von IP auf ATM aussehen kann, und wie u.a. die Adreßabbildung (IP-Adressen auf ATM-Adressen) realisiert werden kann.

ATM-basierte IP-Hosts und -Routers benutzen einen speziellen Server, den MARS, der die Multicastadressen in ATM Link Layer-Adressen auflöst, die dann zur Errichtung von UNIs ,,point-to-multipoint" virtuellen Verbindungen benutzt werden. Über diese Verbindungen werden dann die Nachrichten abgesetzt. Cluster von ATM-Endpunkten werden von einem MARS bedient, der dafür sorgt, daß die einzelnen Verbindungen immer auf den neuesten Stand gehalten werden, da immer wieder neue Prozesse dazustoßen bzw. alte gehen.

Der MARS unterstützt sowohl ein Netz von VCs als auch den Einsatz von Multicastservern, die auf ATM-Ebene arbeiten. Welche Realisierung letztendlich benutzt wird, ist für den Dienstnehmer am ATM-Endpunkt unsichtbar und kann vom Netzwerkadministrator frei gewählt werden.

Broadcasting ist ein Spezialfall des Multicastings und kann ebenfalls durch Einsatz des MARS realisiert werden.

1 Einleitung

Multicasting ist eine sehr universelle Methode der Rechner-Rechner-Kommunikation, da Unicasting und Broadcasting als Spezialfälle angesehen werden können. Besonders bei jeder Anwendung, in der mehrere Prozesse Daten von einer Quelle empfangen müssen, bietet sich der Einsatz von Multicasting an. Bei der heutigen Vernetzung fast aller Einrichtungen unserer Umgebung und den daraus ergebenden Vorteile und Bequemlichkeiten ist meistens die Situation von einem Dienstgeber und einer Masse von Konsumenten gegeben. Diese Empfänger erreicht man am unkompliziertesten per Multicasting. Jeder interessierte Konsument meldet sich in einer Gruppe an, an die die gewünschten Daten gesendet werden.

Im Gegensatz zur Anwendung bestehender Multicastumgebungen ist die Implementation solcher nicht immer eine triviale Angelegenheit. Stehen schon Medien zur Verfügung, die Multicasting ,,von Haus aus" beherrschen (z.B. das Ethernet), ist die Implementierung sicherlich einfacher als bei Kommunikationsmitteln, bei denen ein Sender sich erst anmelden muß und dabei sogar den Adressaten nennen muß, bevor irgendwelche Daten verschickt werden können. Dieses Szenario ist bei ATM-basierten Netzwerken gegeben.

Die folgenden Kapitel beschäftigen sich mit einem Vorschlag der Implementierung von Multicasting auf ATM-Netzwerke.

In Kapitel 2 wird die allgemeine Problematik angesprochen, wie Protokolle, die paketvermittelt arbeiten, auf die ATM Link Layer aufgesetzt werden können. Eine Adaption wird hier am Beispiel von IP erläutert.

Kapitel 3 beschreibt dann die Möglichkeiten, wie Multicasting realisiert werden kann, indem ein Server (der MARS) eingeführt wird, der sowohl für die Abbildung der IP-Adressen auf die ATM-Adressen als auch für die Verwaltung von Multicast-Gruppen verantwortlich ist.

Wie ATM-Endpunkte mit dem MARS interagieren, und welche Protokolle verwendet werden, wird dann in Kapitel 4 besprochen, wobei darauf eingegangen wird, wie sowohl aus der Sicht des MARS als auch der Klienten vorgegangen werden muß.

Einen kurzen Abriß zum Thema Broadcasting und eine Realisierung desselben mittels des MARS wird abschließend in Kapitel 5 gegeben.

2 ATM und IP

Das Problem bei der Implementierung von IP auf einem ATM-Netzwerk besteht darin, daß ATM selbst rein verbindungsorientiert arbeitet. Dazu muß eine Adaptionsschicht geschaffen werden, die paketvermittelnde Dienste zur Verfügung stellt. Der Anschluß eines ATM-Endpunkts geschieht über die vom ATM Forum entwickelte Software UNI 3.0 bzw. 3.1, die verschiedene (u.a. ,,point-to-multipoint" unidirektionale und ,,point-to-point" bidirektionale) Verbindungsdienste anbietet. Das Aufsetzen von IP auf das UNI ist eine nicht-triviale Aufgabe und verlangt unter anderem auch eine Möglichkeit der Adreßauflösung der IP-Adressen auf die Link Layer-Adressen des ATM-Netzwerks. Dazu wird die gesamte Menge der ATM-Endpunkte zu Logical IP Subnets (LIS) zusammengefaßt, die je von einem Address Resolution Protocol (ARP) Server bedient werden (siehe Abbildung 37). Ein solcher Server dient der Zuweisung einer IP-Adresse zur korrespondierenden ATM-Adresse und wird bei jedem Verbindungsaufbau abgefragt. Die Details der Implementierung können im RFC 1577 [Lau93] nachgelesen werden. Die Kombination ATM, UNI und RFC 1577 stellen ein Szenario, welches in Abbildung 38 dargestellt ist. Zwei LIS werden von einem ARP Server bedient, der in diesem Beispiel auch als Router fungiert.

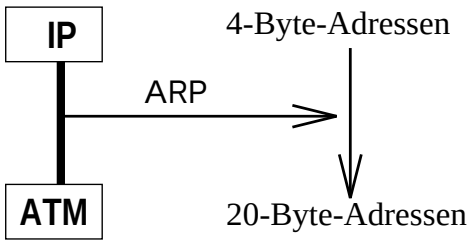

Abbildung 37. ARP bildet IP-Adressen auf ATM-Adressen ab.

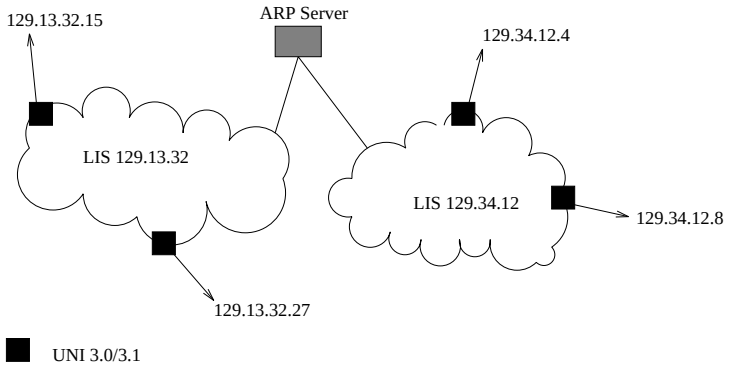

Abbildung 38. Szenario eines ATM-Netzwerks unterteilt in zwei LIS.

3 Multicasting in ATM-Netzwerken unter IP

Die aktuellste Version von UNI stellt die schon oben erwähnten ,,point-to-multipoint" unidirektionalen Verbindungen zur Verfügung, mit denen Nachrichten an Multicast-Gruppen verschickt werden können. Dazu muß eine solche Verbindung vom Sender zu allen Gruppenmitgliedern aufgebaut werden. Das Hinzufügen und Hinwegnehmen von Zweigen aus einer Verbindung kann nur der Initiator durchführen, d.h. er muß immer kontaktiert werden, falls die Gruppe sich ändert.

Unter IP wird der Adressbereich 224.0.0.0 bis 239.255.255.255 für Multicast-Adressen reserviert, d.h. über eine solche Adresse kann eine gesamte Gruppe von Prozessen erreichbar sein. Jeder Prozess muß dazu der zu der Adresse gehörenden Multicast-Gruppe beitreten. Wird nun ein Datagramm an eine Multicast-Adresse geschickt, muß die Möglichkeit gegeben sein, daß die Daten auch alle Mitglieder der Gruppe erreichen. Die IP-Multicast-Implementation (siehe auch RFC 1112 [Dee89]) sieht dazu vor, daß ein an eine Multicast-Adresse gerichtetes Paket nicht geroutet wird, sondern einfach in das lokale Subnetz gespeist wird. Somit empfangen sowohl alle lokalen Gruppenmitglieder die Daten, als auch weiter entfernte, die über einen lokal verfügbaren Multicastrouter versorgt werden.

3.1 Kommunikation zwischen Sender und Empfänger

Solange multicastfähige Medien, wie zum Beispiel das Ethernet, verwendet werden, ist die Umsetzung einer IP-Multicast-Adresse nach Link Layer-Multicast-Adresse trivial und kann meistens algorithmisch gelöst werden. Unter ATM und UNI muß bei der Realisierung auf die schon gegebene Möglichkeit der ,,point-to-multipoint" unidirektionalen virtuellen Verbindungen (VC) zurückgegriffen werden: der Sender eines Datagramms steht an der Wurzel, und die Mitglieder der Gruppe sind die Endpunkte der VC. Dadurch existieren zwei verschiedene Konzepte zur Topologie der VCs:

- Anwendung von Multicastserver (MCS) (Abbildung 39)
- Vermaschung der Endpunkte (VC meshes) (Abbildung 40).

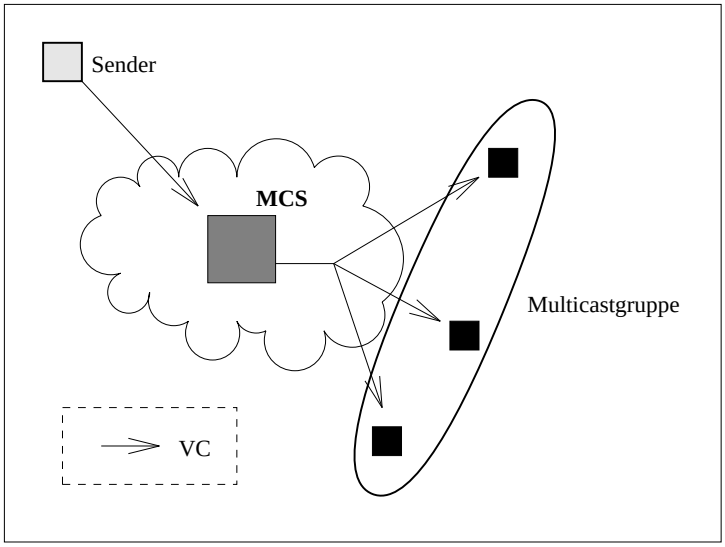

Abbildung 39. Die Gruppe wird über einen MCS erreicht.

Multicastserver empfangen vom sendenden Prozess die gewünschte Zieladresse[2] und das zu versendende Datagramm, welches dann vom Server an alle Mitglieder der Gruppe per unidirektionalem ,,point-to-multipoint" VC weitergeleitet werden. Der Vorteil einer solchen Implementation liegt darin, daß nur eine Verbindung vom sendenden Endpunkt zum ATM-Netzwerk aufrecht erhalten werden muß.

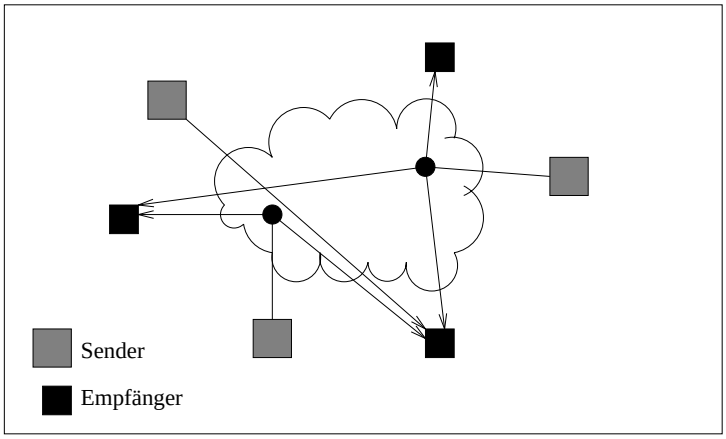

Abbildung 40. Sender und Empfänger sind durch VC-Maschen vernetzt.

VC-Maschen vernetzen den Sender mit allen Mitgliedern der Multicast-Gruppe, wodurch die Kommunikation mit diesen auf direktem Wege geschieht. Dafür muß jedoch für jedes einzelne Gruppenmitglied ein VC gehalten werden, was eventuell höhere Kosten verursachen kann.

[2] Eigentlich wird unter ATM keine Empfängeradresse verwendet. Das Ziel wird durch eine Verbindungsnummer (VPI/VCI) erreicht, die in jeder Zelle mitgeliefert wird

3.2 Auflösung einer Multicast-Adresse

Beiden Realisationsmöglichkeiten ist jedoch das Problem der Auflösung einer Multicast-Adresse gemeinsam. Bei der Betrachtung des ATMARP (das Address Resolution-Protokoll unter ATM) fällt auf, daß man dasselbe Konzept auf Multicast-Adressen erweitern kann. Möchte ein Prozeß Daten an eine Multicast-Adresse schicken, wird bei der Adreßauflösung anstelle einer ATM-Adresse eine ganze Liste von ATM-Adressen geliefert, zu denen dann je ein VC aufgebaut werden kann. Diese Erweiterung des ATMARP Servers stellt der MARS (Multicast Address Resolution Server) dar, der im IETF[3] Internet-Draft von Armitage [Arm96] beschrieben wird. Im weiteren Verlauf werden die Vorgehensweise des MARS und die verwendeten Protokolle beschrieben, ohne zu sehr ins Detail zu gehen.

4 Zusammenspiel von MARS und Klient

Der MARS spielt die zentrale Rolle in der Auflösung von Multicast-Adressen und der Verwaltung der Gruppenzugehörigkeiten. Dazu müssen alle ATM-Endpunkte bei

- der Adressauflösung,
- zum Beitritt in einer Gruppe und
- zum Verlassen einer Gruppe

den MARS kontaktieren. Ebenfalls muß der MARS seine Klienten bei Gruppenänderungen informieren. Es ist wichtig zu verstehen, daß MCSs und Multicastrouter ebenfalls nur Dienstnehmer des MARS sind, und nur geringfügig anders bedient werden.

4.1 Dienste des MARS

Zur Verwaltung der ATM-Endpunkte wird der verwaltungstechnische Begriff Cluster eingeführt. Jeder Cluster wird von mindestens einem MARS versorgt. Zu einem Cluster gehören alle ATM-Endpunkte, die

- sich einen dazugehörigen MARS als Server ausgesucht haben, und
- eine direkte ATM-Verbindung zum MARS aufbauen können.

4.1.1 Verwaltung des Clusters. Bevor ein ATM-Endpunkt Multicasting betreiben kann, muß er einen MARS finden, bei dem er sich registrieren lassen kann. Mit dieser Registrierung wird der Endpunkt in den zum MARS gehörenden Cluster aufgenommen. Nur registrierte Endpunkte werden bedient. Die Registrierung erfolgt durch eine MARS_JOIN-Nachricht[4] an den MARS, der daraufhin den Endpunkt als Blatt in die

[3] Internet Engineering Task Force

[4] Zur Unterscheidung eines JOIN zur Registrierung und des JOIN für den Eintritt in eine Multicast-Gruppe (siehe nächsten Abschnitt) existiert in der Nachricht ein Datenfeld zur Markierung (mar$flags.register).

ClusterControlVC einfügt. Dieser Kanal wird für Kontrollnachrichten vom MARS zu den Klienten benutzt (siehe Abbildung 41). Das Verlassen eines Clusters geschieht durch eine MARS_LEAVE-Nachricht[5], welche zur Konsequenz hat, daß die ClusterControl-Verbindung zum Endpunkt fallen gelassen wird.

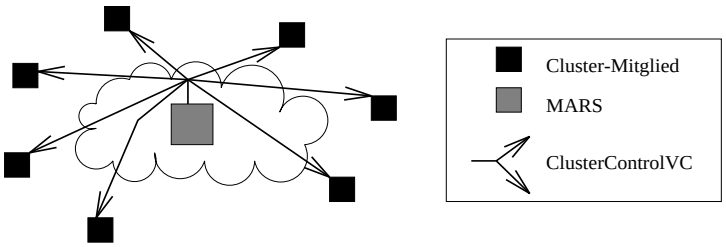

Abbildung 41. MARS und Klienten stehen untereinander in Verbindung.

4.1.2 Verwaltung der Gruppen. In Multicastumgebungen gemäß RFC 1112 geschieht der Beitritt einer Gruppe durch die Nutzung des JoinLocalGroup-Dienstes der IP-Ebene. Durch einen solchen Aufruf in einem MARS-Szenario wird das Senden eines MARS_JOIN an den MARS veranlaßt, der den Endpunkt dann bei sich in die Tabelle in die Liste der zur Gruppe gehörenden ATM-Adressen einträgt. Ähnlich funktioniert das Verlassen einer Gruppe, was z.B. durch ein LeaveLocalGroup bei IP ausgelöst wird. Dazu wird die Nachricht MARS_LEAVE benutzt und der MARS löscht den Endpunkt aus der Gruppe.

Wie schon erläutert wurde, können nur die Eigentümer von VCs zu Multicast-Gruppen diese erweitern bzw. verkleinern, falls sich eine Gruppe ändert. Dazu muß jeder Klient eines MARS ebenfalls von allen MARS_JOINs und MARS_LEAVEs in Kenntnis gesetzt werden, damit die Endpunkte bei Bedarf ihre schon zu einer veränderten Gruppe bestehenden VCs aktualisieren können. Um dies zu erreichen, werden alle JOINs und LEAVEs vom MARS auf dem ClusterControlVC weitergereicht.

4.1.3 Auflösen von Multicast-Adressen. Der MARS verwaltet eine Tabelle, die im Prinzip aus folgenden Zeilen aufgebaut ist:

$$\{\text{Multicastadresse, ATM-Adresse.1, ATM-Adresse.2, } \ldots, \text{ ATM-Adresse.}n\}$$

Diese Tabelle wird durch MARS_JOINs und MARS_LEAVEs immer wieder verändert.

Wenn ein ATM-Endpunkt eine Multicast-Adresse aufgelöst haben möchte, sendet er eine MARS_REQUEST-Nachricht an den MARS und übergibt die Multicast-Adresse. Der MARS sucht nun in der Tabelle nach der Adresse und sendet im Erfolgsfall dem Endpunkt die Liste der ATM-Adressen in einer oder mehreren MARS_MULTI-Nachrichten. Im Falles eines Mißerfolgs wird ein MARS_NAK geliefert.

[5] siehe Fußnote zu MARS_JOIN

4.2 MARS aus der Sicht des Klienten

Auf der Seite des Klienten muß zwischen dem Senden und Empfangen von Multicast-Nachrichten unterschieden werden, da das Senden keinen Beitritt zu einer Gruppe erfordert.

4.2.1 Senden einer Multicast-Nachricht. Bevor ein Datagramm an eine Multicast-Gruppe verschickt werden kann, wird erst einmal kontrolliert, ob der Sender schon einen VC zu der Gruppe bestehen hat. Ist dies der Fall, so ist keine weitere Aktion vonnöten. Ansonsten muß für den Aufbau eines VC die Multicast-Adresse in die Link Layer-Adressen der Gruppenmitglieder aufgelöst werden. Dazu wird dem MARS ein MARS_REQUEST geschickt, der dann in einer oder mehreren MARS_MULTI-Nachrichten die ATM Adressen zurückliefert (siehe Abbildung 42). Zur Sicherung der Übertragung mehrerer dieser Nachrichten werden sie mit Sequenznummern (zweiter Parameter des MARS_MULTI in der Abbildung) versehen, die den Klienten den Verlust einer Nachricht erkennen lassen. Wird ein Sprung in der Sequenz erkannt, wird auf das letzte, zu der Anfrage gehörende MARS_MULTI gewartet und dann ein neues MARS_REQUEST abgesetzt (Abbildung 43). Das letzte MARS_MULTI wird durch ein boolesches Flag gekennzeichnet, welches in der Abbildung der erste Parameter darstellt. Um den Verlust der letzten Nachricht zu erkennen, wird ein Timer benutzt, nach dessen Ablauf der MARS erneut gefragt wird.

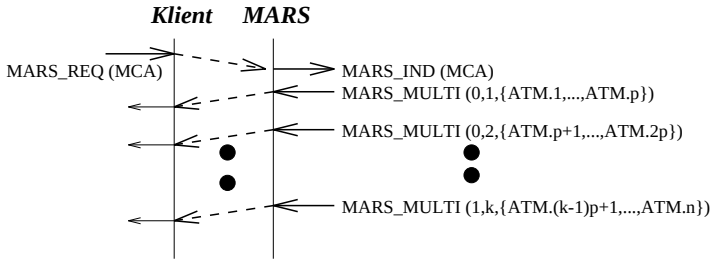

Abbildung 42. Fehlerfreie Auflösung einer Multicast-Adresse.

Sobald die Adressen der Gruppenmitglieder bekannt sind, kann ein ,,point-to-multipoint" VC eingerichtet werden, der den Sender mit jedem Teilnehmer der Multicast-Gruppe verbindet. Es kann durchaus sein, daß eine oder mehrere Verbindungen nicht zustande kommen. In diesem Fall muß nach Fehlermeldung differenziert werden, denn bei temporären Fehlern muß im Hintergrund weiter versucht werden, die Verbindung aufzubauen. Nur bei gravierenden Fehlern wird der ATM-Endpunkt aus der Liste, die vom MARS geliefert wurde, gestrichen.

Über den damit aufgebauten VC kann nun an die gewünschte Multicast-Gruppe gesendet werden.

4.2.2 Aktualisierung der VCs. Da eine Multicast-Gruppe eine dynamische Menge ist, muß auf Seiten der Sender dafür Sorge getragen werden, daß die abgehenden VCs an die Mitglieder einer Gruppe immer auf dem aktuellsten Stand bleiben. Das bedeutet,

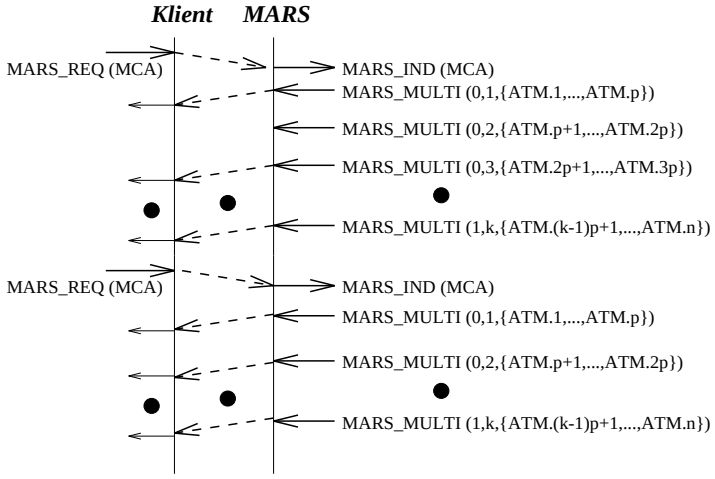

Abbildung 43. Das zweite MARS_MULTI ist verlorengegangen.

daß, sobald ein neuer Endpunkt der Gruppe beitritt, auch gleich ein VC vom Sender mit dem neuen Mitglied der Gruppe aufgebaut wird. Dazu mußsen immer die Nachrichten auf dem ClusterControlVC verfolgt werden. Falls ein neuer ATM-Endpunkt einer Gruppe beitritt, in die ein anderer etwas sendet, so muß der Sender das neue Mitglied in die VC zwischen ihm und der Gruppe aufnehmen. Ebenso muß reagiert werden, falls ein Endpunkt aus einer Gruppe austreten möchte.

4.2.3 Lebensdauer der VCs. Wird ein VC zu einer Multicast-Gruppe für eine längere Zeit nicht mehr benutzt, sollte dafür Sorge getragen werden, daß dieser automatisch abgebaut wird.

4.2.4 Beitreten zu einer Multicast-Gruppe. Das Beitreten zu einer Gruppe geschieht durch das Absetzen einer MARS_JOIN-Nachricht an den MARS unter Angabe der gewünschten Multicast-Gruppe (identifiziert durch ihre Multicast-Adresse). Als Reaktion darauf werden dann im weiteren Verlauf verschiedene potentielle Sender versuchen, einen VC mit dem Endpunkt aufzubauen. Auf diesen VCs können nun die Nachrichten, die an die Gruppe adressiert sind, empfangen werden.

4.2.5 Verlassen einer Gruppe. Durch Senden eines MARS_LEAVE an den MARS (unter Angabe der Multicast-Adresse) wird eine Multicast-Gruppe verlassen. Daraufhin werden alle ehemaligen Sender ihre VCs mit dem Endpunkt abbauen.

4.3 Multicastrouter

RFC 1112 [Dee89] spezifiziert die Funktionsweise von Multicastroutern, die die Aufgabe haben, Multicast-Pakete in andere Subnetze zu leiten. Zu diesem Zweck wird für jedes Subnetz, das der Router bedient, eine Tabelle angelegt, die aussagt, welche Gruppen dort bedient werden müssen (siehe Abbildung 44). Ebenfalls ist er das Ziel von allen im Subnetz abgesetzten Multicast-Nachrichten, da nur er entscheiden kann, ob noch weitere

Gruppenmitglieder in anderen Subnetzen existieren. Im Prinzip bedeutet das, daß der Router Mitglied aller Gruppen ist, die in dem jeweils bedienten Netz benutzt werden. Dies wirft bei Verwendung von ATM und MARS ein Problem auf, da der Router für jede Multicast-Adresse ein MARS_JOIN ausführen müßte, wenn er seinen Dienst aufnimmt.

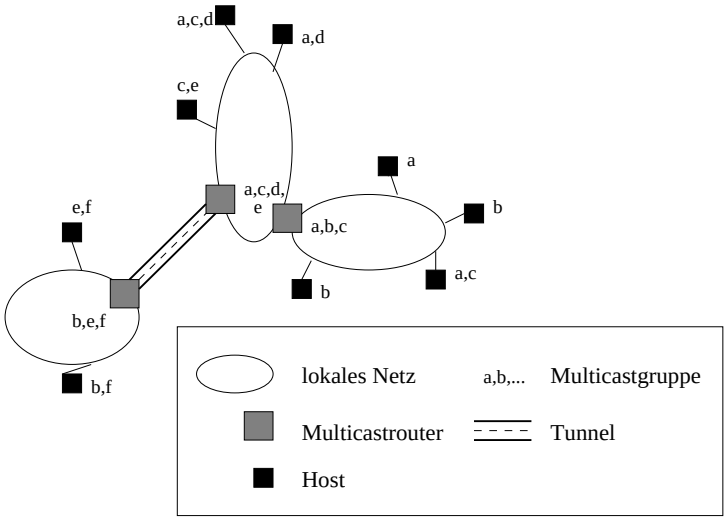

Abbildung 44. Multicastnetze sind durch Multicastrouter verbunden

Das Problem läßt sich dadurch lösen, daß dem MARS bei einem MARS_JOIN nicht nur eine Multicast-Adresse gegeben wird, sondern ein ganzer Bereich von Adressen, spezifiziert durch das Paar $< min, max >$. Ein Multicastrouter braucht somit nur noch die Nachricht MARS_JOIN ($< 224.0.0.0, 239.255.255.255 >$) an den MARS schicken. Einfache Endpunkte dagegen setzen

$$min = max = \text{gewünschte Multicast-Gruppe}$$

und werden für eine Gruppe registriert.

Damit die Router zu jeder Zeit genau Bescheid wissen, welche Gruppen im Subnetz benutzt werden, wurde unter IP das IGMP (Internet Group Management Protocol) integriert. Jeder Aufruf von JoinLocalGroup sendet eine IGMP-Nachricht an 224.0.0.1 (diese Gruppe beinhaltet alle lokalen Rechner) und teilt somit unter anderem den Routern mit, daß Pakete eventuell für eine weitere, neue Gruppe in das Subnetz geleitet werden müssen. Ebenfalls senden die Router in zufällig bestimmten zeitlichen Abständen eine Nachricht an alle lokal verfügbaren Rechner, um die Zugehörigkeiten abzufragen.

Eine solche Vorgehensweise wäre bei Anwendung des MARS jedoch unsinnig, da der MARS selber schon alle für den Router notwendige Daten bereithält. Dazu stellt der MARS das Dienstprimitiv MARS_GROUPLIST_REQUEST bereit, das er dann mit einem MARS_GROUPLIST_REPLY beantwortet, der alle notwendigen Daten beinhaltet. Auf diese Weise wird der Verkehr von IGMP-Nachrichten im ATM-Netz gering gehalten.

Da multicastfähige Netze in einem Netzverbund wie z.B. dem Internet nur kleine Inseln darstellen, müssen diese durch sogenannte Tunnel verbunden werden. Das bedeutet, daß die Multicast-Nachrichten von einem Multicastrouter in eine Unicast-Nachricht

eingepackt werden und diese per Unicast an den Multicastrouter des Empfangsnetzes geschickt werden, wo sie wieder ausgepackt und per Multicast weitergeleitet wird.

4.4 MARS und MCSs

Wie weiter oben schon erläutert wurde, gibt es zu den VC-Maschen auch die Alternative des Einsatzes von Multicastservern (MCS), welche dann die eigentliche Verbindung zu allen Gruppenmitgliedern aufrechterhält (siehe Abbildung 45). Mit ein paar Erweiterungen des schon beschriebenen MARS lassen sich MCSs betreiben, ohne daß die Klienten es bemerken. Bei der Auflösung einer Adresse wird anstelle einer Liste der ATM-Adressen der Gruppenmitglieder nur die Adressen der involvierten MCSs geliefert. Nur wenn ein MCS selber die Mitglieder einer Gruppe erfahren möchte, wird ihm die Liste aller ATM-Adressen der Endpunkte gegeben. Für diese Unterscheidung muß der MARS eine zweite Tabelle, die Servertabelle, verwalten. Multicastserver werden mit der schon oben erwähnten Information

$$\{\text{Multicastadresse, ATM-Adresse.1, ATM-Adresse.2, } \ldots, \text{ ATM-Adresse.}n\}$$

beliefert, während die normalen Endpunkte

$$\{\text{Multicastadresse, MCS-Adresse.1, MCS-Adresse.2, } \ldots, \text{ MCS-Adresse.}m\}$$

erhalten. Um diese Tabelle aufzubauen, muß der MARS von den MCSs wissen, welche Gruppen sie bedienen. Das teilen sie ihm durch die Nachricht MARS_MSERV mit. Wenn ein Server eine Gruppe aus dem Angebot gestrichen hat, muß dies mit MARS_UNSERV dem MARS mitgeteilt werden.

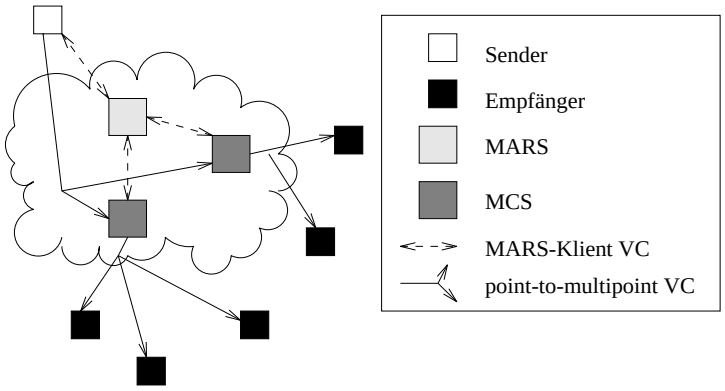

Abbildung 45. VCs zwischen MARS, MCS und Klienten.

Auch MCSs müssen jederzeit auf Änderungen der einzelnen Gruppen hingewiesen werden, welches vom MARS über die ServerControlVC anhand von MARS_SJOIN und MARS_SLEAVE Nachrichten geschieht. Sobald ein MCS sich beim MARS registrieren lässt, wird dieser durch die ServerControlVC mit dem MARS verbunden. Jedes MARS_JOIN bzw. MARS_LEAVE wird über diese Verbindung an alle Server weitergeleitet, die daraufhin ihre VCs zu den Endpunkten aktualisieren.

4.5 Fehlersicherheit

4.5.1 Verlust von Nachrichten. Ein Netzwerk ist niemals gegen Verluste von Daten gesichert. Aus diesem Grunde müssen immer Vorkehrungen getroffen werden, verlorengegangene Nachrichten zu erkennen und nachzufordern. Bei der Kommunikation zwischen MARS und Klienten müssen Verluste in folgenden Situationen erkannt werden:

- Senden der ATM-Adressen per MARS_MULTI,
- Bekanntgeben von MARS_JOINs und MARS_LEAVEs über den ClusterControlVC,
- MARS_SJOINs und MARS_SLEAVEs Meldungen an MCSs (über den ServerControlVC).

In Kapitel 4.2.1 (Senden einer Multicast-Nachricht) wurden schon die Maßnahmen erläutert, die bei einem Verlust von MARS_MULTI Nachrichten getroffen werden.

Sequenznummern werden auch für die Erkennung von JOIN- bzw. LEAVE- Nachrichten benutzt, die nicht empfangen wurden. Dazu wird jeder dieser Nachricht eine clusterweite Sequenznummer (die CSN[6]) zugewiesen, die jedesmal modulo 2^{31} um 1 inkrementiert wird (für SJOINs und SLEAVEs wird eine gesonderte Nummer, die SSN[7], benutzt). Bemerkt ein Endpunkt des Clusters, daß die CSN einer Nachricht auf dem ClusterControlVC nicht den erwarteten Wert (HSN[8]) aufweist, werden sofort alle ausgehenden VCs markiert. Wenn ein Paket durch einen VC geschickt werden soll, der markiert ist, wird, nachdem die Daten verschickt worden sind, der Erneuerungsprozeß gestartet (revalidation). Dies geschieht fast genauso, wie ein neuer VC zu einer Multicast-Gruppe eröffnet wird. Der Unterschied dazu liegt darin, daß, während der VC rekonfiguriert wird, immer noch Nachrichten über denselben verschickt werden können. Eine Erneuerung bedeutet also keine temporäre Schließung eines Kanals. Die dadurch entstehenden Inkonsistenzen müssen von Protokollen höherer Ebenen behoben werden.

4.5.2 Probleme mit dem MARS. Wenn ein Endpunkt einmal keine Antwort auf seine Anfrage vom MARS bekommt, so wird zuerst angenommen, daß es ein transientes Problem ist, d.h. es sollte nach einer gewissen Zeit nochmal versucht werden, den MARS zu kontaktieren. Falls das dann funktioniert, muß der Endpunkt sich wieder registrieren lassen (für den Fall, daß der MARS vorübergehend ausgefallen ist und die Informationen bzgl. der Cluster-Mitglieder verloren gegangen ist). Daraufhin muß der Endpunkt für jede Gruppe, für die er vorher registriert war, ein MARS_JOIN absetzen, bevor dann angefangen werden kann, die einzelnen VCs wieder zu rekonfigurieren. Wichtig ist zu bemerken, daß während der gesamten Prozedur die schon offenen VCs weiter genutzt werden, um den Datenverlust gering zu halten.

Handelt es sich um kein temporäres Problem, sind zwei Alternativen denkbar:

[6] Cluster Sequence Number

[7] Server Sequence Number

[8] Host Sequence Number, die jedes Clustermitglied für sich selbst führt

1. Falls Reserve-MARSs vorhanden sind, muß sich der Endpunkt bei einem von diesen registrieren lassen.

2. Falls keine Reserve existiert, muß der Endpunkt solange in längeren zeitlichen Abständen versuchen, den MARS anzusprechen, bis er entweder von alleine wieder funktioniert oder ein Systemverwalter eingreift.

5 Broadcasting in ATM-Netzwerken

Broadcasting ist im Prinzip ein Sonderfall des Multicastings, bei dem es eine Multicast-Gruppe gibt, die alle Hosts beinhaltet (je nach Reichweite des Broadcasts). Bei Broadcast-Medien, wie z.B. dem Ethernet, ist die Implementierung von Broadcasting eine triviale Angelegenheit. Bei ATM-basierten Netzen muß jedoch eine andere Lösung gefunden werden.

Der MARS, wie er oben beschrieben wird, macht keine Einschränkung bezüglich der Multicast-Adressen, d.h. z.B. unter IP kann auch Gruppen beigetreten werden, deren Adressen nicht im Multicast-Bereich liegen. Somit kann ein MARS zur Realisierung von Broadcasting herangezogen werden, wie es von Smith und Armitage in [SA96] beschrieben wird. Je nach Reichweite des Broadcasts[9] wird einfach ein Multicast an die entsprechende Broadcastadresse gerichtet, die der MARS in ATM-Adressen auflöst. Dazu muß jedoch jeder Host sich beim MARS registrieren lassen und gleich den entsprechenden Multicast-Gruppen beitreten. Unter IP wäre das auf jeden Fall erst einmal 255.255.255.255, bis der Host seine Netzwerk- und Subnetzwerknummer kennt.

Bei der Anwendung eines MARS sollte ebenfalls dafür Sorge getragen werden, daß jedes LIS einen eigenen MARS besitzt, da sonst bei einem beschränkten Broadcast eventuell auch andere Hosts aus einem anderen Subnetz angesprochen werden, weil sie sich unter derselben Gruppenadresse eingetragen haben.

6 Zusammenfassung

Mit Hilfe des MARS ist es möglich, auf ATM-Netzwerken Multicasting zu betreiben. Der MARS dient dabei auf zweierlei Weise:

- für die Auflösung der Multicast-Adressen in ATM Link Layer-Adressen, die für den Aufbau der virtuellen Kanäle vom Sender zur Multicast-Gruppe benötigt
- für die Verwaltung der einzelnen Multicast-Gruppen.

[9] Bei IP kann zwischen

- 255.255.255.255 („all ones" broadcast), alle Hosts
- x.y.255.255, alle Rechner im Netz x und Subnetz y
- x.255.255.255, alle Rechner im Netz x

unterschieden werden.

Sobald ein ATM-Endpunkt Multicast-Nachrichten verschicken möchte, muß dieser sich zu allererst bei einem MARS registrieren lassen. Die Multicast-Adressen werden dann vom MARS in eine Liste von ATM-Adressen abgebildet, die dann für den Aufbau einer Verbindung zu den Gruppenmitgliedern benutzt wird. Änderungen in den Gruppen werden allen beim MARS registrierten Endpunkte bekanntgegeben, so daß die Verbindungen aktualisiert werden können.

Das Beitreten zu bzw. das Verlassen einer Gruppe geschieht ebenfalls durch Senden einer Nachricht an den MARS, der den Endpunkt dann in die Tabellen aufnimmt bzw. streicht.

Multicastserver und Multicastrouter sind bei der Verwendung des MARS transparent für den Dienstnehmer. Die Kommunikation zwischen diesen und dem MARS unterscheidet sich selber nur geringfügig von der normalen Kommunikation zwischen MARS und ATM-Endpunkt.

Broadcasting unter ATM kann durch das Beitreten zu Gruppen mit speziellen Multicast-Adressen mit Hilfe des MARS ermöglicht werden.

Reihenfolgetreue Auslieferung in Multicast-Gruppen

Martin Seeger

Kurzfassung

Innerhalb dieser Ausarbeitung werden Einschränkungen der kausalen und totalen reihenfolgetreuen Multicast-Kommunikationsunterstützung erläutert. Darüber hinaus wird ein Bewertungssystem für reihenfolgetreue Multicast-Protokolle vorgestellt. Um die Einsatzmöglichkeit des Bewertungssystems zu erkunden, werden drei Protokolle unter zur Hilfenahme dieses Bewertungssystems miteinander verglichen. Zusätzlich wird der Fortsetzungsgraf-Algorithmus, der ein weiteres Protokoll für die reihenfolgetreue Multicast-Kommunikation ist, beschrieben.

1 Einleitung

Die Seminararbeit über die reihenfolgetreue Auslieferung in Multicast-Gruppen beinhaltet drei wesentliche Betrachtungsweisen von Protokollen der reihenfolgetreuen Multicast-Kommunikation.

Das zweite Kapitel befaßt sich mit den Einschränkungen der kausalen und totalen reihenfolgetreuen Kommunikationsunterstützung (CATOCS). Es wird zunächst erläutert, was man unter einem kausalen und totalen reihenfolgetreuen Multicast versteht. In diesem Zusammenhang wird die Bedeutung der Begriffe atomar und dauerhaft für diese Multicasts besprochen. Danach werden die vier wichtigsten Einschränkungen des CATOCS und deren Auswirkungen vorgestellt.

Um darüber hinaus verschiedene Lösungen des Problems der reihenfolgetreuen Auslieferung in Multicast-Gruppen miteinander vergleichen zu können, wird im dritten Kapitel ein Bewertungssystem für reihenfolgetreue Multicast-Protokolle vorgestellt. Dieses Bewertungssystem ist in der Lage, netzwerkabhängige Faktoren herauszufiltern und das Ausmaß auftretender Verzögerung zu berechnen. Um die Einsatzmöglichkeit diese Bewertungssystem darzustellen, werden zum Abschluß des Kapitels drei Protokolle innerhalb einer schwach frequentierten Umgebung hinsichtlich ihrer Verzögerung analysiert.

Im vierten und letzten Kapitel wird der Fortsetzungsgraf-Algorithmus vorgestellt. Dieser ist ein weiteres Protokoll für die reihenfolgetreue Multicast-Kommunikation. Es werden in dieser Seminararbeit nur die Hauptmerkmale dieses Algorithmus betrachtet, denn eine detaillierte Bewertung dieses Protokolls sprengt den Rahmen dieser Ausarbeitung.

2 Einschränkungen der kausalen und totalen reihenfolgetreuen Kommunikationsunterstützung (causally and totally ordered communication support, CATOCS)

Bevor die Einschränkungen der kausalen und totalen reihenfolgetreuen Kommunikationsunterstützung erörtert werden, sollte vorab geklärt werden, was man überhaupt unter einer kausalen und totalen reihenfolgetreuen Kommunikation versteht.

In einem kausal reihenfolgetreuen Nachrichtensystem werden Nachrichten in der Reihenfolge übertragen, in der sie vom Sender an den Übertragungsdienst gegeben werden. Die Reihenfolge ist bestimmt durch die „früher geschehen"-Beziehung. Diese Beziehung ist durch den folgenden Sachverhalt charakterisiert: Nachricht m_1 ist „früher geschehen" als Nachricht m_2, wenn m_1 von einem Prozeß zuerst gesendet oder empfangen werden muß, bevor Nachricht m_2 von diesem Prozeß gesendet werden kann.

Das *kausale Multicast* überträgt Nachrichten gemäß der „früher geschehen"-Beziehung innerhalb einer Prozeßgruppe. Besonders wenn zwei Nachrichten an dieselbe Prozeßgruppe per Multicast gesendet werden und das Senden der einen Nachricht früher als das Senden der zweiten Nachricht geschieht, wird die erste Nachricht vor der zweiten Nachricht an alle Prozesse in der Gruppe übertragen. Die Bedeutung eines kausalen Multicasts besteht also darin, daß die „früher geschehen"-Beziehung bewahrt bleibt.

Das *total reihenfolgetreue Multicast* überträgt alle Nachrichten an alle Prozesse innerhalb einer Prozeßgruppe in der gleichen Reihenfolge. Die Nachrichtenübertragung ist gewöhnlich im Bezug der „früher geschehen"-Beziehung zu sehen.

Viele Systeme, auch CATOCS, stellen eine Implementierung der atomaren Nachrichtenübertragung zur Verfügung. Diese atomare Übertragung liefert nur eine Nachricht an alle Prozesse, wenn diese Nachrichtenübertragung an alle Prozesse fehlerfrei abgewickelt werden kann. Ohne atomare Nachrichtenübertragung könnte der Verlust einer Nachricht bei einem Prozeß eine Verzögerung hervorrufen, durch die der Empfang von allen kausal abhängigen Nachrichten nicht mehr bestimmbar wäre. Denn die folgenden Nachrichten, der verlorengegangenen Nachricht, müßten auf unbestimmte Zeit verzögert werden, um die „früher geschehen"-Beziehung zu bewahren. Die Implementierung dieser atomaren Nachrichtenübertragung ist recht einfach. Jeder Prozeß innerhalb der Gruppe puffert jede Nachricht, die er empfängt, bis er sich sicher ist, daß die Nachricht einen stabilen Zustand angenommen hat. Dieser Zustand ist erreicht, wenn z. B. alle restlichen Teilnehmer der Gruppe die Nachricht auch empfangen haben. Bezüglich der atomaren Nachrichtenübertragung, die auch CATOCS zur Verfügung stellt, ist zu erwähnen, daß deren Eigenschaften nicht diejenigen Prozesse einschließen, die während einer CATOCS-Multicast Ausführung versagen. Daher ist die Nachrichtenübertragung zwar atomar, aber nicht dauerhaft.

Definition: Eine Aktion ist dauerhaft, wenn sie trotz ihres Fehlschlagens überlebt, und sie anschließend wieder hergestellt wird.

Die Realisierung einer atomare und dauerhafte Nachrichtenübertragung innerhalb des

CATOCS ist aber im Moment noch zu aufwendig.

Ein allgemeines Problem entsteht bei dem Versuch, CATOCS auf aktuelle Anwendungsprobleme anzuwenden. Dies läßt sich wie folgt charakterisieren: CATOCS ist nicht in der Lage, die Konsistenz der Anwendungsebene zu sichern und einen zusätzlichen Mechanismus für die Zustandsebene bereitzustellen. Um diese Mängel zu beheben, verzichtet man entweder auf die Nutzung des CATOCS, oder die Realisierung wird sehr schwierig.

2.1 Vier spezielle Einschränkungen des CATOCS

Nachdem nun einige allgemeine Problem des CATOCS angesprochen worden sind, werden nun vier spezielle Einschränkungen des CATOCS behandelt.

2.1.1 Nichterkennen der Kausalität

> Kausale Beziehungen zwischen Nachrichten können auf der Semantikebene auftreten. Diese sind durch die „früher geschehen"-Beziehung zwischen Nachrichten nicht zu erkennen und sind daher mittels CATOCS nicht durchsetzbar. Diese Situation kann durch einen externen oder versteckten Kommunikationskanal entstehen, wie z.B. bei einer verteilten Datenbank oder einer externen Umgebung.

Diese Einschränkung vermindert die Einsatzmöglichkeiten der kausalen Kommunikation für viele Anwendungen, denn die meisten Interaktionen finden ausschließlich auf externen Kanälen statt, also außerhalb des Kommunikationsbereichs.

Betrachtet man die kausalen Abhängigkeiten, so kann festgestellt werden, daß sich diese durch das Anfügen der vorgeschriebene Reihenfolgeinformation an Nachrichten leicht handhaben lassen. Diese spiegelt die richtige Reihenfolge und die kausalen Abhängigkeiten wider. Die Empfänger können diese Information nutzen um ihre eigene Ordnung zu sichern. Durch diese Methode wird die Notwendigkeit des CATOCS überflüssig.

2.1.2 Fehlen serieller Fähigkeiten

> CATOCS ist nicht in der Lage, serialisierbare Reihenfolgentreue zwischen Operationen der korrespondierenden Nachrichtengruppen zu sichern.

Updates von Datenstrukturen betreffen in der Regel Gruppen von Speicheroperationen. Ein Update wird benötigt, um die Konsistenz der Anwendungsebene zu gewährleisten. Weiter ist zu bemerken, daß CATOCS nur die Reihenfolgetreue für die individuellen Nachrichten liefert. Ein Update, das eine Menge von Nachrichten verlangt, wird als eine serielle Einheit gehandhabt. Diese serielle Einheit benötigt einen zusätzlichen Mechanismus, der die Notwendigkeit des CATOCS überflüssig macht.

2.1.3 Die nicht angesprochenen semantischen reihenfolgetreuen Einschränkungen

> Viele semantische reihenfolgetreue Einschränkungen können von der „früher geschehen"-Beziehung nicht ausgedrückt werden und sind daher von CATOCS nicht durchsetzbar.

Verallgemeinert man die vorangegangene Einschränkung, so sieht man, daß das korrekte Verhalten einer Anwendung innerhalb ihrer Zustände reihenfolgetreue Restriktionen über die Operationen verlangt. Diese Retriktionen beziehen sich auf die semantischen reihenfolgetreue Einschränkungen und sind stärker als die reihenfolgetreuen Einschränkungen, die durch die „früher geschehen"-Beziehung auferlegt wurden. Dies hat zur Folge, daß geringfügige semantischen reihenfolgetreuen Einschränkungen, wie es z.B. beim kausalen Gedächtnis der Fall ist, durch das Benutzen eines totalen reihenfolgetreuen Multicast erzwungen werden können. Diese Restriktionen können aber nicht durch den Gebrauch eines kausalen Multicast durchgesetzt werden. Die Realisierung der totalen reihenfolgetreuen Multicast-Protokolle ist sehr aufwendig. Man realisiert deshalb die semantischen reihenfolgetreuen Einschränkungen durch einfachere Protokolle, welche die logische Uhren der Zustandsebene nutzen.

Zur Realisierung von stärkeren reihenfolgetreue Einschränkungen, welche liniearisierbar und serialisierbar sind, ist kein kausaler und kein totaler reihenfolgetreue Multicast ausreichend.

2.1.4 Kein Leistungsvorteil gegenüber der Technik der Zustandsebene

> CATOCS Protokolle haben keine Leistungsvorteile gegenüber der Technik der Zustandsebene und scheinen weit weniger skalierbar zu sein.

Innerhalb des CATOCS besitzt jede Nachricht einen Overhead zur Wahrung der Reihenfolge. Dieser hebt aber dennoch nicht die Notwendigkeit der vorgeschriebenen Reihenfolge der Nachrichten und die geforderten Operationen für die ënd to endSSemantik auf. Das heißt, CATOCS beseitigt oder reduziert z. B. nicht den Bedarf von Zeitmarken oder die Mehrfachversionen innerhalb der Echzeitanwendungen. Außerdem neigt CATOCS dazu, auf Grund einer falschen Kausalität, die Nachrichten zu verzögern. Eine falsche Kausalität tritt immer dann auf, wenn Nachrichten auf der Kommunikationsebene einen zufälligen kausalen Zusammenhang haben, jedoch dieser Zusammenhang zwischen den Nachrichten keine semantische kausale Abhängigkeit ist. Diese Situation entsteht dadurch, daß die „früher geschehen"-Beziehung zwischen Nachrichten nur einen Hinweis auf eine mögliche Kausalität liefert. Diese Beziehung ist aber nicht in der Lage, eine tatsächliche Kausalität zu signalisieren. Das bedeutet also nicht notwendigerweise, daß – falls eine Nachricht vor einer zweiten empfangen wird – die erste die zweite verursacht haben muß. Die falsche Kausalität reduziert die Leistungsfähigkeit deswegen, weil die Nachrichten solange unnötige verzögert werden, bis die frühste, vermeintlich „kausal verwandte", Nachricht empfangen wird.

Insgesamt beschränkt sich das CATOCS als Kommunikationseinrichtung auf die Zusicherung der Semantik auf der Kommunikationsebene. Das heißt, CATOCS kann keine Anwendung der Zustandsebene oder keine „end-to-end“ Semantik-Anforderungen erkennen und erzwingen. Diese Gesamteinschränkung spiegelt das „end-to-end“ Argument wider, welches besagt, daß eine Einrichtung auf niedriger Ebene nicht die Semantiken auf den höheren Ebenen zusichern kann. Sie kann bestenfalls eine Optimierung für den „high level“ Mechanismus sein.

Nun kennt man die wichtigsten Einschränkung eines CATOCS. Möchte man nun verschiedene Protokolle der reihenfolgetreuen Multicast-Kommunikation miteinander vergleichen, benötigt man ein Bewertungssystem, womit man die einzelnen Protokolle analysieren kann. Solch ein Bewertungssystem für reihenfolgetreue Multicast-Protokolle soll im nächsten Abschnitt vorgestellt werden.

3 Ein Bewertungssystem für reihenfolgetreue Multicast-Protokolle

Innerhalb dieses Kapitels wird ein Bewertungssystem für reihenfolgetreue Multicast-Protokolle vorgestellt. Danach wird die Einsatzmöglichkeit des Bewertungssystems anhand von drei Protokollen überprüft.

3.1 Existierende Bewertungsmethoden

Um erst einmal einen Einblick über existierende Bewertungsmethoden zu bekommen, werden die wichtigsten Bewertungssysteme kurz vorgestellt.

Reihenfolgetreue Multicast Protokolle werden oft hinsichtlich der Anzahl an Kontrollnachrichten verglichen. Diese werden erzeugt, um Benutzernachrichten übertragen zu können. Christian et al. [CA86] berechnet analytisch die durchschnittliche Anzahl solcher Nachrichten, die per Broadcast durch eine Kommunikationskette gesendet wurden. Chang und Maxemchuk [CM84] berechnen die Anzahl der Kontrollnachrichten pro Benutzer-Broatcast der unterschiedlichen Protokollfamilien. Die Autoren legen dabei eine Broadcast-Netzwerkumgebung zugrunde. So wird innerhalb ihrer Bewertungsmethode jede Broadcastoperation als eine einfache Kontrollnachricht berechnet. Garcia-Molina und Spauster rechnen mit der Wahrscheinlichkeit, daß die Broadcast-Einrichtungen nicht zur Verfügung stehen. Für diesen Fall, berechnen sie jeden Broadcast wie n Kontrollnachrichten, wobei n die Anzahl der Empfänger ist. Außerdem können reihenfolgetreue Multicast-Protokolle hinsichtlich der Bearbeitungszeit bewertet werden. Christian mißt die Multicast-Leistung innerhalb eines Zeitabschnitt, welcher definiert ist als ein Zeitintervall zwischen dem Senden einer Nachricht und ihrer Übertragung an die letzte Empfängergruppe.

3.2 Modell, Architektur und dazugehörende Funktionen

Innerhalb des Abschnitt 3.2 wird ein reihenfolgetreues Multicast als ein Dienst definiert. Außerdem wird eine Architektur vorgestellt, die ein reihenfolgetreues Multicast-Protokoll

beschreibt. Für diese Architektur werden Funktionen definiert, mit denen man in der Lage ist, die Protokolle hinsichtlich der Verzögerung zu bewerten.

Um die Funktion des Nachrichtenaustausch besser erläutern zu können, wird folgendes angenommen: Eine Anwendung, die die Multicast Kommunikation nutzt, besteht aus mehreren Sendern S und Empfängern R mit:

$$S = \{S_1, S_2, \ldots S_m\}$$
$$R = \{R_1, R_2, \ldots R_n\}$$

Die Sender und Empfänger kooperieren beim Austausch von Nachrichten über eine Menge

$$Q = \{q_1, q_2, \ldots\}.$$

Der Nachrichtenaustausch wird von mehreren zuverlässigen Multicast-Operationen vollzogen. Jede Nachricht von Q stammt von genau einem Sender S. Außerdem wird von Q genau eine Nachricht zu jedem Empfänger R übertragen. Beschreibt man diesen Vorgang formal, so definiert man eine Eintrittsfunktion I mit:

$$I = \begin{cases} Q \to R \\ q \mapsto I(q) = \text{Zeit, zu der } q \text{ von einem Sender übergeben wurde} \end{cases}$$

und eine Übertragungsfunktion D mit

$$D = \begin{cases} Q \to R^n \\ q \mapsto (D^1(q), D^2(q), \ldots, D^n(q)) & \text{mit } D^i(q) \text{ die Zeit, die benötigt wird, um die Nachricht an den Empfänger } R_i \text{ zu übertragen} \end{cases}$$

Dabei sind beide Funktionen von Q abhängig, und es gilt $D^i(q) > I(q) \forall q \forall i$.

Die Sender arbeiten unabhängig voneinander und können ihre Multicast-Nachrichten parallel ausgeben. Diese Eigenschaft wird „gleichzeitiges Multicast-Szenario“ genannt, wenn es eine semantische Abhängigkeit zwischen den ausgegeben Multicasts der verschiedenen Sender gibt, die eine Synchronisation der individuellen Multicast notwendig macht.

Das reihenfolgetreue Multicast ist eine spezielle Form einer gleichzeitigen Multicast-Synchronisation. Zwei beliebige Nachrichten zweier Sender müssen von allen Empfänger in der gleichen Reihenfolge empfangen werden, um die globale Semantik einzuhalten. Dieser Sachverhalt kann durch die folgenden Notation ausgedrückt werden.

$$\forall q1, q2 \in Q \forall 1 \leq i, j \leq n : \quad [D^i(q_1) < D^i(q_2)] \Rightarrow [D^j(q_1) < D^j(q_2)]$$

Gemäß den verschiedenen Netzwerkverzögerungen werden die Multicast-Nachrichten, falls keine Synchronisation stattfindet, zu verschiedenen Zeitpunkten bei den Empfängern

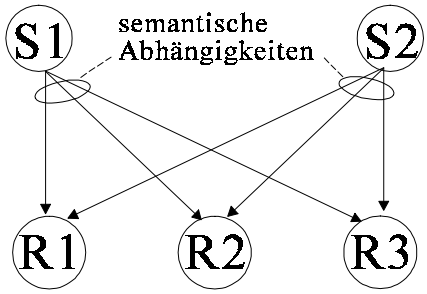

Abbildung 46. Multicast-Szenario

eintreffen. Dies hat zur Folge, daß Multicast-Nachrichten von verschieden Sendern in unterschiedlichen Reihenfolgen an die individuellen Empfänger übertragen werden. Um dies zu verdeutlichen, wird nun ein Szenario betrachtet, bei dem ein Sender S_1 und ein Empfänger R_1 durch ein LAN verbunden sind, und ein Sender S2 mit einem Empfänger R_2 über ein zweiten LAN verbunden sind. Dies kann nur, wie man auf dem Abbildung 47 sehen kann, mittels eines WANs erreicht werden.

Eine Multicast-Nachricht q1, die von einem Sender S_1 innerhalb einer globalen Zeit $t = 0$ ausgegeben wurde, kommt auf der Empfängerseite R_1 zum Zeitpunkt $t = 20$ und zum Zeitpunkt $t = 100$ auf der Empfängerseite $R2$ an (gemäß der Größe der WAN-Verzögerung). Eine andere Multicast-Nachricht q_2, die zur selben Zeit von einem Sender S_2 gesendet wurde, wird als erstes auf der R_2-Seite ankommen (Abbildung 47).

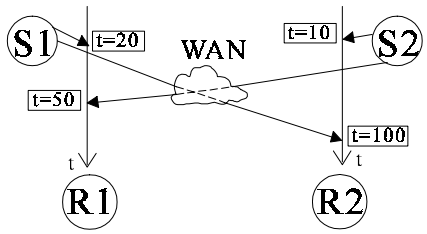

Abbildung 47. Nachrichtenlaufzeiten bedingen eine unterschiedliche Auslieferungsreihenfolge bei verschiedenen Empfängern

Wenn das System q_1 und q_2, sofort nachdem sie an der jeweiligen Stelle angekommen sind, überträgt, z. B. $D(q_1) = (20, 100)$ und $D(q_2) = (50, 10)$, wäre die reihenfolgetreue Multicast-Eigenschaft somit verletzt. Verlangt man nun eine totale Ordnung für dieses Beispiel, so könnte ein reihenfolgetreues Multicast-Protokoll die Übertragung von q_1 an R_1 bis zum Zeitpunkt $t = 51$ verzögern und könnte in der Zwischenzeit die anderen Nachrichten sofort ausliefern.

3.2.1 Die Synchronisationsschicht Innerhalb einer speziellen Architektur kann das reihenfolgetreue Multicast einer unabhängigen Synchronisationsschicht zugewiesen werden, die unterhalb der Anwendungs- und über der Transportschicht des Netzwerkes angesiedelt ist. Die Transportsicht stellt zwar einen zuverlässigen Multicast-Transfer zur Verfügung, aber dieser ist nicht synchronisiert. Anwendungen benutzen deshalb einen

Transportdienst, der um die Funktionalität zur Synchronisation ergänzt wurde. Innerhalb der Multicast-Synchronisationsschicht befinden sich zwei Typen von Nachrichten- und Schnittstellenprimitiven:

- Die eine wird für den Transfer der Benutzerdaten benötigt.
- Die andere benutzt diesen Transfer zur Kontrolle, z.B. zur Synchronisation der simultanen Übertragungsinstanzen.

Zwei Unterkomponenten der Schicht verdeutlichen diese Einteilung.

Die Benutzer-Datenübertragung Die Benutzernachrichten müssen so geordnet werden, daß sie an allen Empfangsstellen in der gleichen Reihenfolge ankommen. Dies hat zur Folge, daß die Benutzerdaten solange verzögert werden müssen, bis die vorrangigen Benutzernachrichten eintreffen. Aus diesem Grund muß eine Verzögerungseinheit zur Verfügung stehen, die die ankommenden Benutzernachrichten in einen lokalen Speicher puffert, so daß diese zu einem späteren Zeitpunkt unverändert übertragen werden können. Um die Verzögerung einer Nachricht bestimmen zu können, muß ihre Ankunft von einer Beobachtungseinheit registriert und geordnet werden. Hierzu wird eine Beobachtungs/Verzögerungskomponte (Observe/Delay, O/D) benötigt.

Synchronisationsübertragung Die O/D-Komponente stellt ein Werkzeug für die Kontrolle der Benutzernachrichten dar. Diese O/D-Komponente wird durch die Synchronisations-Dienst-Einheit (Synchronization Service Entity, SSE) überwacht. Die SSE ist für die Koordination der Übertragungen verantwortlich. Innerhalb jeder Empfangs- und Sendestelle existiert eine solche SSE. Diese SSE besitzt eine Schnittstelle zur Tranzportschicht und eine Schnittstellen zur O/D-Komponente (Abbildung 48).

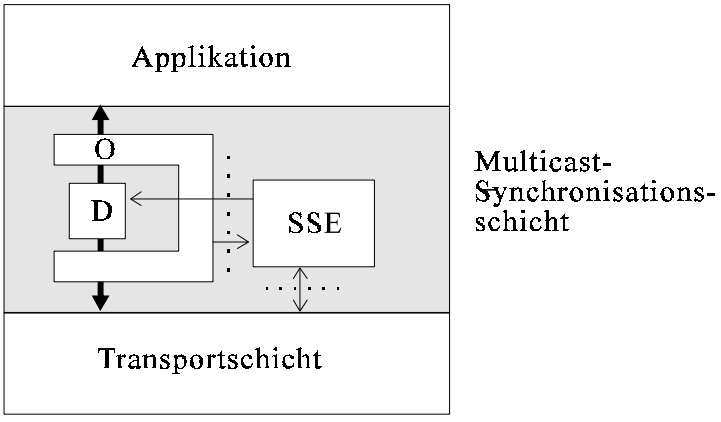

Abbildung 48. Das der Bewertung zugrundeliegende Schichtenmodell

Über die Transportschichtschnittstelle tauschen die SSEs untereinander Protokolldaten aus. Die SSEs führen alle ein Mehrpartner-Protokoll aus, um die Übertragung der Nachrichten zu koordinieren (Abbildung 49).

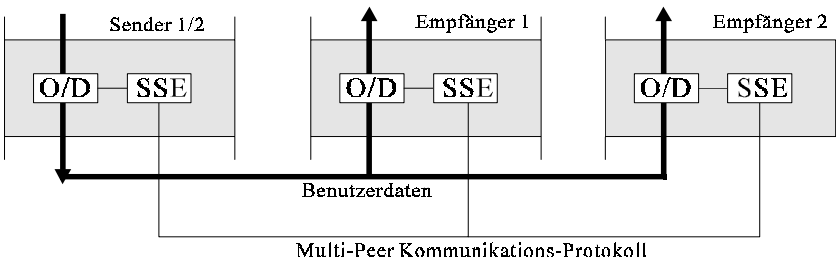

Abbildung 49. Austausch von Protokolldaten

Die O/D-Schnittstelle einer SSE befindet sich innerhalb der Multicast-Synchronisationsschicht und vereinigt die Elemente des Benutzerdaten-Transfers mit den Elementen der Synchronisation. Diese Vereinigung geschieht folgendermaßen: Trifft eine Nachricht, von der Transportschicht kommend, bei der O/D-Komponente ein, so signalisiert das Beobachtungselement der O/D-Komponente das Eintreffen der Nachricht der SSE. Die SSE wiederum sendet eine bestimmte Nachricht zurück an die O/D-Komponente, die in dem Verzögerungselement der O/D-Komponente gepuffert wird. Diese Aufteilung der Multicast-Synchronisationsschicht in die Komponenten O/D und SSE ist dazu bestimmt, die Abwicklung der Nachrichten, die während der Protokollausführung stattfindet, zu strukturieren.

Diese Architektur kann in folgenden Punkten verändert werden:

- Protokolldaten der SSE benutzen dieselben Dienst-Zugangspunkte auf der Transportschicht wie der Request der Benutzerdaten. In diesem Falle muß ein Selektor in der Nachricht eine korrekte Zuordnung der Nachrichten erlauben.
- Protokolldaten von einer SSE können an Nachrichten der Benutzerdaten angehängt werden (piggyback), um so das Protokoll zu optimieren.

3.3 Herleitung der Synchronisationsverzögerung

Es wurde bereits erwähnt, daß bei Forderung einer reihenfolgetreuen Semantik die Verzögerung der Benutzernachrichten innerhalb der O/D-Komponenten nicht vermieden werden kann. Es zeigt sich anhand der Summe der Verzögerung, daß die Zeit, um die Benutzernachrichten verzögert werden, stark von der Entscheidungen des Synchronisations-Protokoll abhängt.

Anstatt die Verzögerung des Netzwerks zu messen, berechnet man den Verzögerungs-Overhead, der von einem reihenfolgetreuen Protokoll verursacht wird, das nicht die optimale Sequenz wählt.

Es werden dazu folgende Annahmen gemacht:

a) Der Nachrichtenaustausch an den Schnittstellen und die Abwicklung verzögern die Nachrichten. Die Verzögerung auf dem Netzwerk und die Verzögerung, die durch die O/D-Komponente verursacht wird, sind die einzigen Faktoren, die zur Berechnung benötigt werden.

b) Das Mehrpartner-Multicast Protokoll, das auf allen SSEs läuft, hat keine Auswirkung auf die Verzögerungen der Benutzernachrichten auf den Netzwerk.

c) Die Auslieferung der Nachrichten zu den Anwendungen wirkt sich nicht auf die Übertragung weiterer Nachrichten durch die Anwendung aus.

d) Die O/D-Komponenten der Sender verursachen keine Verzögerungen.

Die zweite Annahme verkörpert eine wichtige Beschränkung, denn hier könnten die Nachrichten eines Multicast Synchronisations-Protokolls zu einer Verstopfung des Netzwerkes führen. Zu erwähnen ist in diesem Zusammenhang, daß zwar die Synchronisationsnachrichten generell von ihrer Größe kleiner sind als die Benutzernachrichten, jedoch sollte ihr Einfluß auf den Gesamtverkehr dennoch begrenzt sein.

Zur Beschreibung der Beeinflussung des Netzwerkes wird eine Netzwerk-Verzögerungsfunktion N, kurz N-Verzögerung definiert.

$$N = \begin{cases} Q \to R^n \\ q \mapsto (N^1(q), N^2(q), \ldots, N^n(q)) \end{cases} \quad \begin{array}{l} \text{mit } N^i(q) \text{ die Netzverzögerung der Nachricht } q, \\ \text{die an den Empfänger } R_i \text{ gesendet wird} \end{array}$$

Die N-Verzögerung beschreibt die Zeit, die eine Nachricht zwischen dem Verlassen der Multicast-Synchronisationsschicht des Senders und dem Eintritt an der Empfängerseite benötigt. Falls die Verzögerung von Sender S_i zu Empfänger R_j immer konstant und unabhängig von der Richtung ist, so werde die Verzögerung mit δ_{ij} bezeichnet. Es gilt dann:

$$\forall q \forall j: \quad [S_i \text{ gab } q \text{ aus}] \Rightarrow [N^j(q) = \delta_{ij}]$$

Die Definition einer Ankunftsfunktion A lautet:

$$A = \begin{cases} Q \to R^n \\ q \mapsto (A^1(q), A^2(q), \ldots, A^n(q)) \end{cases} \quad \text{mit } A^i(q) = I(q) + N^i(q)$$

Innerhalb dieser Architektur ist A als Zeitpunkt definiert, zu dem eine Nachricht von der Transportschicht zu der O/D-Komponente des Empfängers gelangt. Eine weitere abgeleitete Funktion ist die Synchronisations-Verzögerungsfunktion Δ, kurz S-Verzögerung. Sie mißt den Aufenthalt einer Nachricht in einer O/D-Komponente.

$$\Delta = \begin{cases} Q \to R^n \\ q \mapsto \Delta(q) = D(q) - A(q) \end{cases}$$

Zuletzt wird die Summe der S-Verzögerung $\overline{\Delta}$ definiert:

$$\overline{\Delta} = \sum_{q \in Q} |\delta(q)| \quad \text{mit} \quad |\delta(q)| = \sum_{i=1}^{n} \Delta^i(q)$$

Die S-Verzögerungssumme beschreibt die Summe der Synchronisationsverzögerung. Mit diesen Verzögerungs-Funktionen können nun Protokolle bewertet werden.

3.4 Das Verhalten der reihenfolgetreuen Multicast-Protokolle bei geringem Datenverkehr

Der Abschnitt 3.4 erläutert die Einsatzmöglichkeit des Bewertungssystem anhand dreier Protokolle. Es wird das Verhalten dieser Protokoll bei geringem Datenverkehr analysiert.

Unter Berücksichtigung der Frage, wie sich die Protokolle bei schwachem Multicast-Verkehr verhalten, werden hier drei reihenfolgetreue Multicast-Protokolle hinsichtlich ihrer Leistung untersucht und miteinander verglichen. Die Untersuchung berücksichtigt, im Falle daß andere Multicast-Nachrichten noch gesendet werden, die geringe Wahrscheinlichkeit für das Senden einer Multicast-Nachricht. Das charakteristische Verhalten eines Protokolls unter geringem Datenverkehr kann durch Untersuchung der Auswirkungen eines einzelnen Sender-Pilot beschrieben werden. Ein einzelner Sender-Pilot geht davon aus, daß nur eine einzige Nachricht von einem willkürlichen Sender ausgegeben wird. Nach dem Prinzip des einzelnen Sender-Pilots sind auch die Protokolle konstruiert, die nun vorgestellt und analysiert werden.

3.4.1 Ein Sender-kontrolliertes Protokoll Zuerst wird ein Sender-kontrolliertes Protokoll vorgestellt. Das ABCAST Protokoll von Birman/Joseph [Bir87] ist ein solches Sender-kontrolliertes Protokoll: Die SSEs der sendenden Teilnehmer spielen bei der Bestimmung der Übertragungsreihenfolge eine koordinierende Rolle (Abbildung 50).

Um ein solches Protokoll besser erläutern zu können, verfolgt man die verschiedenen Nachrichtenübertragungen, die in Abbildung 50 dargestellt sind. Die O/D-Komponente des Senders verzögert keine Nachrichten. So kann eine Benutzernachricht (1) sofort an die Transportschicht gegeben werden, welche sie per Multicast an die Empfänger sendet (2). Die O/D-Komponenten der Empfänger signalisieren ihren jeweiligen SSEs den Empfang dieser Nachricht (3) und puffern diese in einen lokalen Speicher. Jede SSE der Empfänger bestimmt die Priorität dieser Nachricht und sendet diese Priorität mittels der Transportschicht an die SSE des Senders zurück (4). In der zweiten Phase kommt die Koordinationsrolle der SSE des Senders zum tragen. Sie berechnet das Maximum dieser Prioritäten und sendet diese per Multicast zurück an die SSEs der Empfänger (5). Die SSEs veranlassen (6) nun ihre O/D-Komponente die Benutzernachricht auszuliefern (7). Der Gebrauch von Prioritäten garantiert, daß alle Empfänger-SSE dieselbe Entscheidung für die Wahl der nächsten zu übertragenden Nachricht treffen.

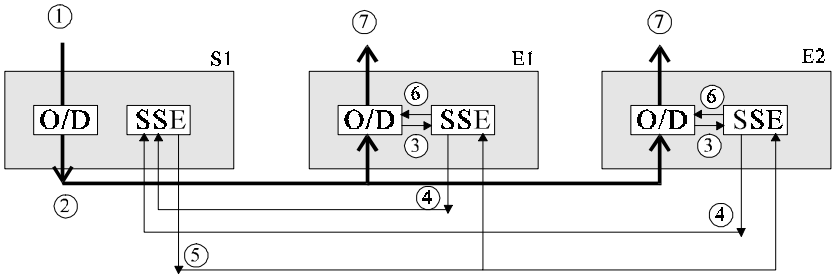

Abbildung 50. Nachrichtenaustausch beim Sender-kontrollierten Protokoll

Die Ankunftsfunktion A für das Sender-kontrollierte Protokoll ist definiert als:

$$A(q_1) = (\delta_{11}, \delta_{12}, ..., \delta_{1n})$$

Die Zeitmarken (Priorität) der Nachrichten werden sofort von den SSEs der Empfänger erzeugt und kommen zurück zum Sender S1. Dieser Vorgang nimmt folgende Zeit in Anspruch: $2\delta_{11}$, $2\delta_{12}$, usw. Bis zum Empfang der letzten Nachricht vergeht folgende Zeit:

$$t = \max_{i=1...n} \{2\delta_{1i}\}$$

Die letzte zu übertragene Synchronisationsnachricht ist demnach der Multicast zu den Empfänger. Für die gesamte Sender-kontrollierte Protokollausführung erhält man die folgende Übertragungsfunktion:

$$D^i(q) = \max_{j=1...n} \{2\delta_{1j}\} + \delta_{1i}, \quad i = 1, 2, \ldots, n$$

Berechnet man das Ergebnis der Differenz $D(q_1) - A(q_1)$ innerhalb einer S-Verzögerungssumme, so bekommt man

$$\overline{\Delta} = 2 \cdot n \cdot \max_{j=1...n} \{\delta_{1j}\}$$

Nimmt man an, daß die N-Verzögerung zwischen jeweils zwei Standorten gleich δ ist, dann lautet die S-Verzögerung

$$\overline{\Delta} = 2 \cdot n \cdot \delta.$$

3.4.2 Ein Empfänger-kontrolliertes Protokoll Das zweite Protokoll, das nun vorgestellt wird, ist ein Empfänger-kontrolliertes Protokoll. Das Protokoll von Chang und Maxemchuk [CM84] gehört z.B. dieser Klasse an. Die SSEs auf der Empfangsseite koordinieren die Übertragung selbst. Das bedeutet, die Nachrichtenübertragungen finden ohne Unterstützung eines Senders statt (Abbildung 51). Treffen nun Nachrichten beim Empfänger ein (2), wird dies jeder SSE signalisiert. Einer der Empfänger wird zum „Hauptempfänger“ bestimmt. Im Gegensatz zu den anderen Empfänger kann er sofort die Übertragung der Nachrichten (3) anordnen. Gleichzeitig sendet der Hauptempfänger eine Bestätigung mit der Reihenfolge, in welcher die Nachrichten übertragen werden müssen, zu den anderen Empfänger (4). Die „Nebenempfänger“ ordnen die Auslieferung einer Benutzernachricht (5,6) erst dann an, wenn sie die Benutzernachricht vom Sender und die Bestätigung vom „Hauptempfänger“ haben.

Für die Berechnung der Verzögerung dieses Protokolls wird angenommen, daß R_1 der Hauptempfänger ist und das Netzwerk die Nachricht zwischen R_1 und den restlichen Empfänger R_1, R_2, ..., R_n durch δ_1, δ_2, ..., δ_n verzögert. Somit kann man die Übertragungsfunktion für dieses Protokoll aufstellen.

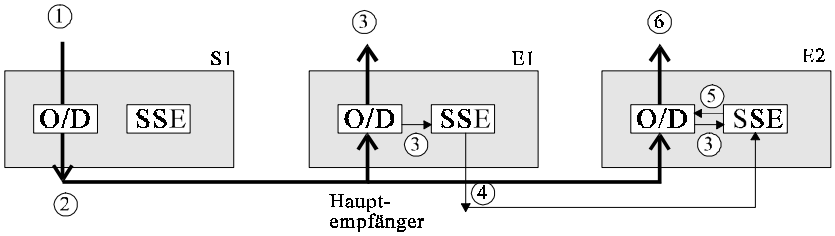

Abbildung 51. Nachrichtenaustausch beim Empfänger-kontrollierten Protokoll

$$D^1(q_1) = A^1(q_1) = \delta_{11}$$
$$D^i(q_1) = \max\{\delta_{1i}, \delta_{11} + \delta_i\} \quad \text{für} \quad i = 2, 3, \ldots, n$$

Dieses führt zu der S-Verzögerungssumme

$$\overline{\Delta} = \sum_{i=2}^{n} \max(0, \delta_{11} + \delta_i - \delta_{1i})$$

Für eine identische N-Verzögerung δ zwischen Teilnehmern im Netzwerk bekommt man eine einfache Formel: $\overline{\Delta} = (n-1) \cdot \delta$.

3.4.3 Ein Zeit-kontrolliertes Protokoll Das dritte Protokoll, das hier vorgestellt werden soll, ist ein Zeit-kontrolliertes Protokoll. Das Protokoll von Christian et al. [CA86] benutzt eine Gesamtdifferenz, die Grundlage für die Zeit-kontrollierte Methode ist. Diese Methode kann angewendet werden, wenn Netzwerk und Kommunikationssoftware Echzeit-Eigenschaften besitzen, insbesondere wenn es eine obere Zeitschranke für eine „end-to-end“ Übertragungsnachricht gibt. Unter dieser Bedingung stellt das Zeit-kontrollierte Protokoll eine geordnete Übertragung mit wenigen Synchronisationsnachrichten zur Verfügung (Bild 7). Die SSE eines Senders fügt jeder Benutzernachricht (1) eine Synchronisationsnachricht (3) an (2), beide werden zu den Empfängern gesendet. Die SSE-Nachrichten enthalten eine Zeitmarke, die von einer Uhr erzeugt wird, die innerhalb der Sender SSE verwaltet wird. Die SSEs der Empfänger leiten von diesem Wert, zusammen mit der Netzwerklatenz und den Gangunterschieden der Uhren, den letzten Zeitpunkt her, bei dem alle Benutzernachrichten auf der Empfängerseite sein müssen. Diese Zeit ist als Übertragungszeit definiert. Jede SSE der Empfänger verwaltet eine lokale Uhr (4) und ordnet zu diesem Zeitpunkt die Übertragung der Benutzernachricht an. Da alle Empfänger eine Nachricht zur gleichen Zeit übertragen, und da die Uhrzeit an allen Empfangsstellen monoton wächst, werden die Benutzernachrichten überall in der identischen Reihenfolge übertragen (5).

Das Zeit-kontrollierte Protokoll führt zu einer S-Verzögerung, denn die Nachrichten an jeden Empfänger werden so lange verzögert, bis der letzte diese Nachrichten empfangen hat. Die Übertragungsfunktion für dieses Protokoll ist definiert als:

$$D^i(q_1) = \varepsilon + \max_{j=1\ldots n} \{\delta_{1j}\} \quad i = 2, 3, \ldots, n,$$

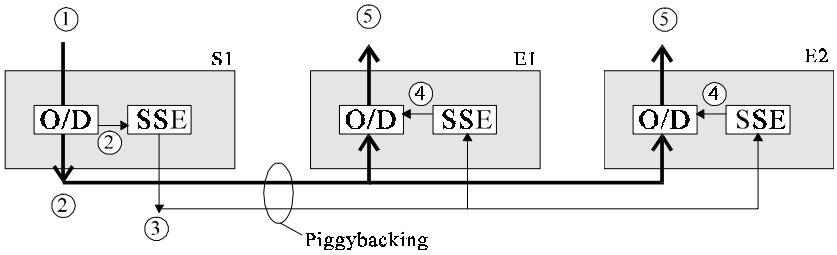

Abbildung 52. Nachrichtenaustausch beim Zeit-kontrollierten Protokoll

wobei ε die Synchronisationsvarianz der Uhren ist.

Die S-Verzögerungssumme berechnet sich demnach wie folgt:

$$\overline{\Delta} = n\varepsilon + n \cdot \max_{i=1\ldots n}\{\delta_{1i}\} - \sum_{i=1}^{n} \delta_{1i}$$

Für identische Verzögerungen und vollständig synchronisierte Uhren gilt: $\overline{\Delta} = 0$.

Die Tabelle liefert eine Zusammenfassung der durchschnittlichen S-Verzögerung.

Einzelner Sender Pilot $(\overline{\Delta}/n)$	Sender-kontrolliert	Empfänger-kontrolliert	Zeit-kontrolliert
Allgemeiner Fall	$2 \max_{i=1\ldots n}\{\delta_{1i}\}$	$\frac{1}{n}\sum_{i=2}^{n} \max(0, \delta_{11} + \delta_i - \delta_{1i})$	$\varepsilon + \max_{i=1\ldots n}\{\delta_{1i}\} - \frac{1}{n}\sum_{i=1}^{n} \delta_{1i}$
Identische N-Verzögerung, Synchronisierte Uhr	2δ	$\frac{n-1}{n}\delta$	0

Tabelle 1. Durchschnittliche S-Verzögerung

Durch die Ergebnisse der Bewertung von Protokollen ist der Benutzer eines Systems in der Lage, geeignete Protokoll für seine Anwendung auszuwählen.

4 Fortsetzungsgraf-Algorithmus

Das vierte und letzte Kapitel dieser Seminararbeit beschäftigt sich mit einem Protokoll für die reihenfolgetreue Multicast-Kommunikation. Dieses Protokoll wird in der Literatur Fortsetzungsgraf-Algorithmus genannt. Der Fortsetzungsgraf-Algorithmus löst vielfache reihenfolgetreue Multicast- Gruppen Probleme durch eine bestimmte Organisation logischer Knoten innerhalb eines Waldes (Grafentheorie). Bevor man den Fortsetzungsgraf-Algorithmus erklärt, sollten die Probleme der reihenfolgetreuen Multicast-Kommunikation berücksichtigt werden. Das Hauptproblem und die Hauptaufgabe eines Multicast-Protokolls ist es, die reihenfolgetreue Übertragung von Nachrichten zu den Zielprozessen zu garantieren. Zum besseren Verständnis werden die verschiedenen Arten der Reihenfolgetreue aufgezählt.

a) *Einfach reihenfolgetreue Quelle.* Wenn die Nachrichten m_1 und m_2 von demselben Sender erzeugt werden und sie an dieselbe Multicast-Gruppe adressiert sind, dann bekommen alle Zielprozesse diese Nachrichten in derselben Reihenfolge.

b) *Mehrfache reihenfolgetreue Quelle.* Wenn die Nachrichten m_1 und m_2 an dieselbe Multicast-Gruppe adressiert sind, dann bekommen alle Zielprozesse diese Nachrichten in derselben Reihenfolge, auch wenn sie von verschieden Quellen stammen.

c) *Mehrfach reihenfolgetreue Gruppe.* Wenn die Nachrichten m_1 und m_2 an zwei Prozesse übertragen werden, werden sie in derselben Reihenfolge übertragen, auch wenn sie von verschiedenen Quellen stammen und an verschiedene, aber überlappende Multicast-Gruppen adressiert sind.

Es gibt Anwendungen, die diese Eigenschaften nicht verlangen. Aber es gibt natürlich auch Anwendungen, bei denen der Empfang von Nachrichten in unterschiedlicher Reihenfolge zu Inkonsistenz und zu Totpunkt-Problemen führen kann.

Um diese Inkonsistenz zu vermeiden, wurde das Protokoll des Fortsetzungsgraf-Algorithmus entwickelt. Man entwarf ein Modell, in dem Nachrichten innerhalb eines Netzwerk und an alle Knoten fehlerfrei übertragen werden. Innerhalb des betrachteten Netzwerks werden Nachrichten zwischen einem Knotenpaar in einer bestimmten Reihenfolge übertragen. Ein Ziel der Struktur des Systems ist es zu versuchen, einige der Overheads der gesamt Lösungen zu reduzieren. Die Nachrichten werden durch mehrere strukturierte Knoten innerhalb des Fortsetzungsgraf angeordnet. Jeder Knoten des Grafen repräsentiert dabei eine Berechnungsstelle. Der Graf beschreibt die Pfade, auf denen die Nachrichten ihr Ziel erreichen. Die Nachrichten werden somit über mehrere Knoten weitergereicht. Diese Knoten ordnen die Nachrichten entlang des Pfades, indem sie die Nachrichten für verschiedene Gruppen miteinander verschmelzen. Die Knoten sind damit in der Lage, die richtige Reihenfolge der empfangenen Nachrichten zu erkennen. Diese Tatsache hat zur Folge, daß die Knoten nur noch auf die früher abgeschickten Nachrichten warten müssen, bevor sie die schon empfangenen Nachrichten an den Zielprozeß übertragen können.

Die Grundidee ist daher, daß man die Knoten als Vermittlungsknoten innerhalb der Schnittpunkte der Multicast-Gruppen verwendet.

Der Fortsetzungsgraf-Algorithmus besteht aus dem Fortsetzungsgraf (propagation graph, PG) Generator und dem Nachrichtenübermittlungsprotokoll (message passing, MP). Der PG-Generator ist verantwortlich für den Aufbau des Fortsetzungsgrafen für eine gegebene Menge von Multicast-Gruppen. Das MP-Protokoll wird von einem Knoten genutzt um Nachrichten zu senden, zu empfangen und weiter zu geben.

4.1 Der PG-Generator

Betrachtet man den PG-Generator näher, so sollte er eine Garantie für folgende Eigenschaften liefern:

1. Alle Nachrichten werden in derselben Reihenfolge übertragen.

2. Wenn x sich innerhalb einer Gruppe α befindet, bekommt x alle Nachrichten, die für die Gruppe α bestimmt sind.

Um diese Forderungen zu erfüllen, ist es ausreichend, wenn man für den Fortsetzungsgrafen folgende Eigenschaften betrachtet.

(PG1) Für jede Gruppe α gibt es nur ein primäres Ziel p.

(PG2) Für jede Stelle $x \in \alpha$ gibt es nur einen Pfad von p nach x.

Es gibt außerdem zwei optionale Eigenschaften, die der Graf aufweisen kann. Der PG-Generator versucht, diese Eigenschaften bereitzustellen:

(PG3) Das primäre Ziel der Gruppe α ist in der Gruppe α enthalten.

(PG4) Ist p das primäre Ziel von α, und ist x ein anderer Knoten der Gruppe α, dann sind alle Knoten, die auf dem Pfad von p nach x liegen, auch in der Gruppe α.

Um den Algorithmus des PG-Generator besser erläutern zu können, zeigt Abbildung 53 ein Beispiel für einen Fortsetzungsgrafen. Dieser besteht aus neun Knoten: a, b, c, d, e, f, g, h und j und aus acht Zielgruppen: $\alpha_1 = \{c, d\}$, $\alpha_2 = \{a, b, c\}$, $\alpha_3 = \{b, c, d, e\}$, $\alpha_4 = \{d, e, f\}$, $\alpha_5 = \{e, f\}$, $\alpha_6 = \{b, g\}$, $\alpha_7 = \{c, h\}$ und $\alpha_8 = \{d, j\}$.

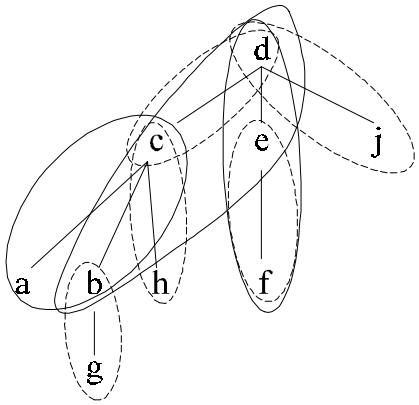

Abbildung 53. Beispiel für einen Fortsetzungsgrafen

Der PG-Generator bestimmt aus den Zielgruppen denjenigen Knoten, der in den meisten Gruppen enthalten ist, und macht ihn zur Wurzel. Im obigen Beispiel ist dies der Knoten d. Diese Heuristik hilft, den Baum in einem Wald klein zu halten. Zum besseren Verständnis nennt man die Gruppe, in der sich die Wurzel befindet, Wurzelgruppe. Die Wurzel ist das erste Ziel aller Wurzelgruppen. Die Kinder der Wurzel werden durch eine Prozedur *new_subtree* bestimmt. Die Arbeitsweise der Prozedur ist folgende: Zuerst zerlegt sie die „Nicht-Wurzelgruppen“ in mehrere disjunkte Mengen. In dem obigen Beispiel erhält man somit zwei disjunkte Mengen: $P_1 = \{a, b, c\}, \{b, g\}, \{c, h\}$ und $P_2 = \{e, f\}$. Von diesen Mengen wird eine zum Kind der Wurzel gewählt. Nun beginnt die gleiche Prozedur von vorn. Man wählt wieder den Knoten, der in den meisten Untermengen enthalten ist,

und macht ihn zur Unterwurzel. Die Prozedur wird so lange rekursiv aufgerufen, bis alle Knoten abgearbeitet sind. Durch diese Rekursion wird ein baumförmiger Graf aufgebaut (siehe Abbildung 54). Für die Knoten, die nach dem Prozedur *new_subtree*-Aufruf noch nicht behandelt worden sind, muß eine neue Wurzel gewählt werden.

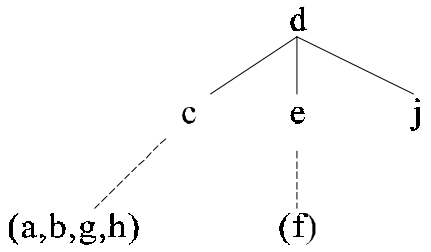

Abbildung 54. Der Beispielgraf nach einigen Arbeitsschritten des Algorithmus.

4.2 Das MP-Protokoll

Der Fortsetzungsgraf spezifiziert den Fluß der Nachrichten innerhalb eines Netzwerkes. Die Wurzel beinhaltet das primäre Ziel jeder Multicast-Gruppe. Empfängt ein Knoten eine Nachricht, die sich im Teilbaum nach unten fortpflanzt, so ist der Knoten Teil einer Zielgruppe für diese Nachricht. Im obigen Beispiel ist d das primäre Ziel für $\{c, d\}$,{b, c, d, e}, $\{d, e, f\}$ und $\{d, j\}$. Empfängt nun die Instanz d eine Nachricht für $\{b, c, d, e\}$, so sendet sie eine Kopie zu c und e. Die Nachricht wird auf diese Weise durch den Baum nach unten gehangelt. Um die Bearbeitung der Nachrichten innerhalb des MP-Protokolls zu verdeutlichen, sei im folgenden eine Analyse für ein Punkt-zu-Punkt Netzwerk dargestellt. Das MP-Protokoll verlangt von jedem Knoten, für jeden Knoten, der ihm eine Nachricht sendet, Sequenznummern zu führen. Die Sequenznummern werden vom dem Fortsetzungsgraf bestimmt. Dies garantiert, daß ein Empfänger die Nachrichten eines Senders, die möglicherweise ungeordnet eintreffen, korrekt anordnen kann. Diese Methode kann auch zur Entdeckung von verlorengegangenen Nachrichten verwendet werden.

4.2.1 Warteschlangen innerhalb des MP-Protokolls Innerhalb des MP-Protokoll sind keine Empfangsbestätigungen nötig. Sendepausen werden zur Fehlerentdeckung genutzt. Eine weitere Charakteristik dieser Methode ist, daß jeder Knoten zwei Warteschlangen besitzt. Es gibt eine Warteschlange für die Nachrichten, die für einen lokalen Prozeß bestimmt sind, und es gibt eine Warteschlange für die Nachrichten, die in der falschen Reihenfolge ankommen. Empfängt nun ein Knoten eine Nachricht, überprüft er die Sequenznummer mit der Sequenznummer, die er von dem Sender erhalten hat. Stimmen diese nicht miteinander überein, wird die Nachricht in einer Warteschlange des Knoten abgelegt. Die Nachricht verharrt solange in der Warteschlange, bis die vor ihr ausgelieferte Nachricht eingetroffen ist. Stimmen die Sequenznummern überein, wird der nachfolgende Empfänger bestimmt, für den die Nachricht bestimmt ist. Der Empfänger sendet danach die Nachricht an seine Kinder, die wiederum Unterwurzeln des Baumes sind. Ist der Empfänger in einer Zielgruppe enthalten, so stehen die Nachrichten für eine

lokale Auslieferung in der Warteschlange. Zusätzlich überprüft der Empfänger, ob es in der Warteschlage Nachrichten vom Sender gibt, auf die er gewartet hat. Diese Nachrichten werden auf dieselbe Art und Weise wie oben erläutert behandelt.

Somit sind die wichtigsten Komponenten und die Funktionsweise des Fortsetzungsgraf-Algorithmus beschrieben, und es könnte nun eine Leistungsanalyse durchgeführt werden. Diese Analyse wird in dieser Arbeit nicht erläutert. Es soll aber dennoch erwähnt werden, daß der Fortsetzungsgraf-Algorithmus in vielen Fällen Vorteile gegenüber den Senderkontrollierten Protokollen hat. Die Nachteile dieser Methode liegen bei dem großen Implementierungsaufwand des Fortsetzungsgrafen, deshalb sollte diese Technik nur bei langlebig Multicast-Strömen benutzt werden.

5 Zusammenfassung

Im zweiten Kapitel wurden Einschränkungen des CATOCS vorgestellt. Es wurden die vier wichtigsten Einschränkungen herausgearbeitet. Dabei hat man gesehen, daß das Nichterkennen der Kausalitäten, das Fehlen serieller Fähigkeiten, die nicht angesprochenen semantischen reihenfolgetreuen Einschränkungen und der fehlende Leistungsvorteil gegenüber der Technik der Zustandsebene innerhalb des CATOCS die Anwendung des CATOCS sehr einschränken. Das Anwendungsgebiet des CATOCS beschränkt sich somit auf die Zusicherung der Semantik auf der Kommunikationsebene. Es wurden deshalb andere Mechanismen entwickelt, die die reihenfolgetreue Kommunikation auf höheren Ebenen gewährleisten. Diese Mechanismen wurden aber in dieser Ausarbeitung nur kurz erwähnt.

Im dritten Kapitel wurde eine neue Methode für die Bewertung der reihenfolgetreuen Multicast-Protokolle erläutert. Innerhalb einer Multicast-Sychronisationsarchitektur wurde die Synchronisationsverzögerung definiert, die zur Ermittlung der Gesamtverzögerung bestimmt wurde. Diese Synchronisationsverzögerung ist das Herzstück dieses Bewertungssystems, das den Vergleich zwischen reihenfolgetreuen Multicast-Protokolle ermöglicht. Um die Einsatzmöglichkeit dieses Systems zu zeigen, wurde das Verhalten von drei Protokollen bei geringem Multicast-Verkehr analysiert. Das vierten Kapitel stellt den Fortsetzungsgraf-Algorithmus vor. Dieser Algorithmus ist ein entwickeltes Protokoll für reihenfolgetreue Multicasts. Es wurde in diesem Kapitel nur auf die Funktionsweise des Algorithmus eingegangen, denn eine Betrachtung der Leistung dieses Protokolls hätte den Rahmen dieser Seminararbeit gesprengt.

Aspekte verteilter Simulationen

Frank Lamprecht

Kurzfassung

Der folgende Beitrag beschäftigt sich mit Simulationen und den Problemen bei deren Verteilung auf ein Rechnernetzwerk. In den unterschiedlichsten Gebieten werden Simulationen eingesetzt, um das Verhalten komplexer Systeme zu untersuchen, verstehen oder optimieren zu können. Je aussagekräftiger aber die Ergebnisse der Simulationen sein sollen, um so langwieriger sind die Berechnungen. Um die Geschwindigkeit der Berechnungen zu steigern, wird versucht, die Simulationen auf mehrere Prozessoren oder Rechner zu verteilen. Die verschiedenen Aspekte, die dabei beachtet werden müssen, sollen im folgenden erläutert werden. Nach einer Einführung der Prinzipien von verschiedenen Simulationsmethoden mit besonderem Augenmerk auf ereignisorientierte Simulationen werden Lösungsansätze zur Parallelisierung durch Verteilung dargestellt.

1 Was ist Simulation?

Bei einer Simulation wird das Verhalten eines realen Systems durch ein Modell nachgebildet. Solch ein Modell ist in der Regel ein vereinfachtes Abbild des komplexen realen Systems, verhält sich aber bezüglich der relevanten Aspekte genauso wie dieses.

Es gibt verschiedene Gründe, weshalb man bestrebt ist, ein reales System durch ein Modell nachzubilden. Beispielsweise ist das reale System zu gefährlich, um damit Versuche anzustellen, wie dies bei einem Reaktor der Fall wäre. Weiterhin könnte es zu kostspielig sein, ein reales System für Versuchszwecke zu benutzen, wie zum Beispiel bei Crashtests. Auch die Zeit spielt eine entscheidende Rolle. Wollte man die Entwicklung der Weltbevölkerung für das nächste Jahrhundert bestimmen, dauerte es zu lange, 100 Jahre zu warten. Andererseits läuft zum Beispiel die Zündung in einem Verbrennungsmotor zu schnell ab, um sie untersuchen zu können.

Somit ist es also wünschenswert oder auch notwendig, manche Vorgänge zu simulieren, um die realen Gegebenheiten optimieren, testen oder überprüfen zu können.

Natürlich beinhalten Simulationen auch Probleme. Ein sehr großes Problem ist zum Beispiel die tatsächliche Realitätsnähe. Es ist in der Regel unmöglich, ein reales System vollständig und mit allen Gesetztmäßigkeiten nachzubilden. Somit sind die Simulationsergebnisse mit Vorsicht auszuwerten, da sie schwer zu bestimmende Ungenauigkeiten enthalten. Ein weiteres sehr großes Problem ist die Modellbildung und Simulation auf dem Computer. Wenn man ein System auf dem Computer modellieren möchte, ist man schon durch den verfügbaren Speicher und die Möglichkeiten der Simulationssoftware in dem Grad der Realitätsnähe beschränkt. Hinzu kommt dann noch die Laufzeit der Simulation. Die Rechner werden zwar immer leistungsfähiger, doch ist die Komplexität eines Systems meist auch unbeschränkt groß, und man wird versucht sein, das Modell so genau wie möglich zu erstellen. Neben der Verwendung immer leistungsfähigerer Computer wird deshalb ein Weg gesucht, höhere Leistung mit der vorhandenen Hardware zu

erzielen. Dies ist möglich durch die Vernetzung mehrerer Rechner und der Verteilung der Simulation auf diese Rechner. Hier bieten sich auch interessante Möglichkeiten mit dem Internet. Bei der Verteilung auf viele Rechner, die über das Internet verbunden sind, darf natürlich der Kommunikationsbedarf zwischen den Rechnern nicht sehr groß sein.

Derartige Rahmenbedingungen werden aber oft schon durch die Art und Weise bestimmt, wie ein reales System in eine Simulation überführt wird. Welche Möglichkeiten es hier gibt, soll das nächste Kapitel darlegen. Anschließend wird darauf eingegangen, welche Möglichkeiten der Verteilung der verschiedenen Arten von Simulationen existieren.

2 Die Arten der Simulation

Simulationen lassen sich in zeitgesteuerte und ereignisgesteuerte Systeme untergliedern. Prinzipiell kann man bei zeitgesteuerten Simulationen noch zwischen kontinuierlichen und diskreten differenzieren. Da jedoch für die Berechnung auf dem Computer alles diskretisiert werden muß, sind die kontinuierlichen Simulationen nur annäherbar.

Bei der Umsetzung von realen Systemen in Computersimulationen ist ein wichtiges Merkmal zu beachten: In der Realität brauchen die Zustandsänderungen, die simuliert werden sollen, selbst keine Zeit, da sie sich normalerweise kontinuierlich ändern und nur zu bestimmten Zeitpunkten beobachtet werden. Dafür vergeht unter Umständen sehr viel Zeit zwischen den Beobachtungszeitpunkten. Bei der Computersimulation dagegen ist diese Zeit vernachlässigbar und die Zustandsänderungen müssen in der Regel mit großem Zeitaufwand berechnet werden. Dieses Dualitätsprinzip [MM89] ist sehr wichtig bei den Betrachtungen zur Verteilung und parallelen Verarbeitung von Simulationen, wie später noch gezeigt wird.

Zunächst sollen die beiden Grundarten der Simulation erläutert werden, wobei der ereignisgesteuerten die größere Bedeutung zukommt, da die meisten Simulationen nach diesem Prinzip ablaufen. Der im folgenden verwendete Begriff der Simulationszeit bezieht sich nicht auf die Laufzeit der Berechnungen, sondern auf die virtuelle Zeit innerhalb der Simulation.

2.1 Zeitgesteuerte Simulation

Die zeitgesteuerte Simulation könnte man auch als quasi-kontinuierliche Simulation bezeichnen. Hierbei wird immer nach einem vorgegebenen Zeitintervall der aktuelle Zustand des Systems aus dem vorhergehenden Zustand berechnet. Beispiele für kontinuierliche Systeme, die man hiermit modellieren kann, finden sich vor allem im physikalisch-technischen Bereich, z.B. Wettersimulationen. Derartige Systeme werden oft durch Differentialgleichungen beschrieben, so daß die Aufgabe des Computers in einer numerischen Lösung dieser Gleichungen besteht.

Ein weiteres, wenn auch sehr einfaches, Beispiel einer zeitgesteuerten Simulation (ohne Differentialgleichungen) ist Conway's Game of Life. Hier bestimmen einfache Gesetzmäßigkeiten den jeweils nächsten Zustand. Es wird eine Zellpopulation simuliert, wobei einfach die Positionen eines zweidimensionalen Feldes entweder mit einer oder

keiner Zelle besetzt sind. Bei dieser Simulation wird davon ausgegangen, daß nach einer bestimmten Zeitspanne Zellen entsprechend der Anzahl ihrer Nachbarn starben, neu entstanden oder unverändert blieben. Die entsprechende Zeitspanne kann bei der Simulation übergangen werden. Rechenzeit kostet jedoch das Zählen der Nachbarn usw.

Die Dauer der Zeitspanne zwischen zwei Beobachtungs- bzw. Berechnungszeitpunkten ist im allgemeinen ein sehr wichtiger Punkt. Sie spielt eine entscheidende Rolle bei der Gesamtlaufzeit der Simulation und auch bei der Aussagekraft der Resultate. Wählt man die Zeitspanne zwischen zwei Berechnungszeitpunkten sehr groß, läuft die gesamte Simulation sehr schnell ab. Dadurch können aber unter Umständen wichtige Resultate verlorengehen. Dies verhält sich ganz entsprechend dem Abtasttheorem: Mit einer festgelegten Abtastrate kann man nur Frequenzen bis zu einer bestimmten Größe erfassen. Würde man bei der Simulation andererseits die Zeitspanne zwischen den Berechnungen sehr klein wählen, wird unter Umständen viel Rechenzeit verschwendet. Indem man eine sehr kleine Zeitspanne wählt, erhält man eine Annäherung an eine kontinuierliche Simulation.

2.2 Ereignisgesteuerte Simulation

Bei der ereignisgesteuerten Simulation finden die Berechnungen nicht nach vorbestimmten Zeitabschnitten statt, sondern nur dann, wenn bestimmte Ereignisse eintreten. Ist ein Ereignis eingetreten, werden die Folgen daraus berechnet. Dies sind in der Regel auch immer weitere Ereignisse, die in der Zukunft liegen. Diese resultierenden Ereignisse werden im allgemeinen in einer Ereignisliste gespeichert, die nach den Eintrittszeitpunkten geordnet ist. Anschließend wird die Simulationszeit auf den Eintrittszeitpunkt des nächsten Ereignisses gesetzt und dieses bearbeitet.

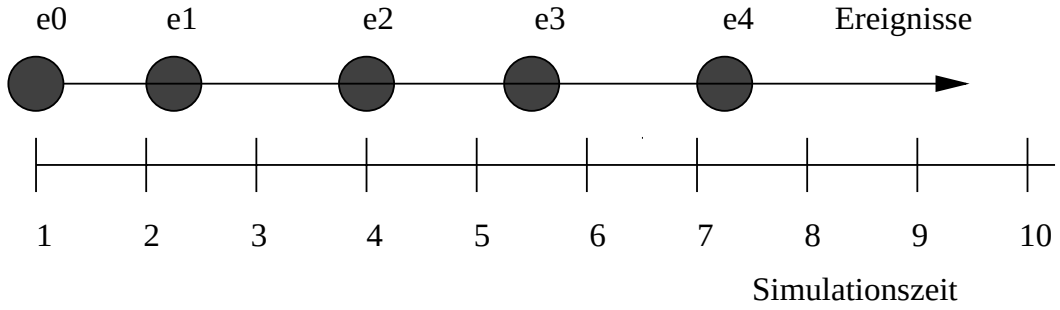

Abbildung 55. Schematische Darstellung einer Ereignisliste.

Man kann sich das ganze am besten an einem Beispiel verdeutlichen. Betrachtet man ein Billardspiel, gibt es hier als Ereignisse die Kollisionen von Kugeln. Das initiale Ereignis ist das Anstoßen der weißen Kugel. Aus Anstoßwinkel und -kraft und den Positionen der anderen Kugeln kann man das nächste Ereignis, die Kollision mit einer Kugel auf dem Weg und dessen Eintrittszeitpunkt berechnen. Die Zeit, in der die Kugel nun unterwegs zu dieser Kollision ist, kann übersprungen werden, da hier nichts passiert.

Hier wird auch ein Problem der zeitgesteuerten Simulation deutlich, würde man diese Simulationsart benutzen. Wählt man nämlich die Zeitabstände zu groß, wird die Kugel

womöglich immer gleich um mehrere Zentimeter weitergesetzt und verfehlt ein Kollision vollständig. Bei der ereignisgesteuerten Simulation wiederum wird die Simulationszeit nun einfach auf den Zeitpunkt der nächsten Kollision gesetzt. Da nach der Kollision in der Regel nun zwei Kugel sich bewegen, können zwei neue Ereignisse berechnet und in die Ereignisliste eingetragen werden. Das ganze setzt sich solange fort, bis keine Ereignisse mehr in der Liste stehen. Es gibt noch viele weitere Beispiele für ereignisgesteuerte Simulationen, wie zum Beispiel die Simulation einer Ampelsteuerung. Hierbei wären die Ereignisse das Eintreffen von Autos an der Ampel und das Umschalten der Ampelphasen.

Es gibt natürlich noch viele Variationen der ereignisgesteuerten Simulation. Bei dem oben beschriebenen Billardspiel gibt es nur eine globale Ereignisliste. Eine oft praktizierte Variante ist, die Simulation aus einer Menge interagierender Objekte aufzubauen. Hier ist auch schon ein Ansatz von Parallelität zu sehen, auf welchen später noch eingegangen wird. Die Objekte beim Beispiel Billard wären die Kugeln. Jede Kugel hat dann eine eigene Ereignisliste, in der die eigenen Kollisionen eingetragen werden. Da bei einer Kollision aber immer mindestens zwei Kugeln beteiligt sind, muß solch ein Ereignis immer auch bei einem anderen Objekt in die Ereignisliste eingetragen werden. Eine ruhende Kugel hat zum Beispiel zu Beginn eine leere Ereignisliste. Wenn nun eine andere Kugel in Richtung dieser ruhenden Kugel gestoßen wird, trägt die angestoßene Kugel als Objekt die folgende Kollision bei sich selbst und bei der ruhenden Kugel in die Ereignisliste ein.

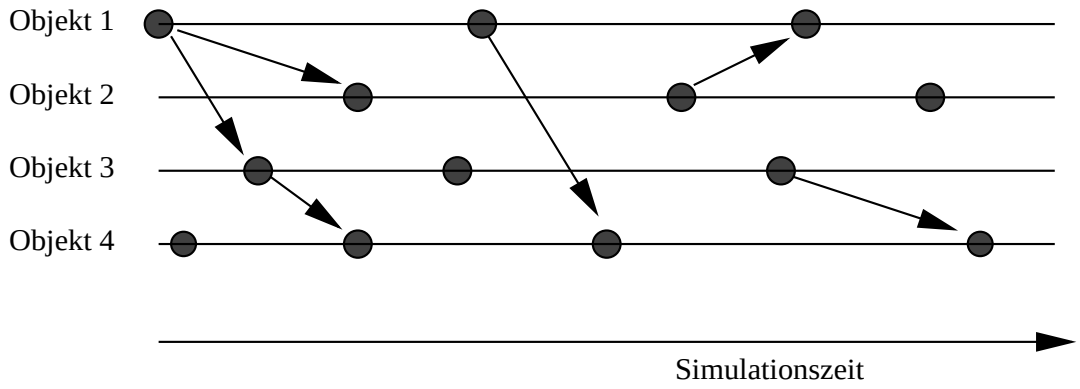

Abbildung 56. Allgemeines Beispiel für vier parallel arbeitende Objekte.

Genau dieses Eintragen in fremde Listen bereitet Probleme bei der Parallelisierung. In der Realität läßt sich oftmals solch eine Einteilung in Objekte vornehmen, die offensichtlich gleichzeitig nebeneinander agieren. Zur Parallelisierung könnte man nun einfach jedem Prozessor solch ein Objekt zuteilen. Der Prozessor bearbeitet dann jeweils ein Ereignis und springt zum nächsten in der Ereignisliste und bearbeitet dieses. Das Problem ist nun, daß er zwischen zwei solchen Ereignissen noch ein weiteres eingetragen bekommen könnte. Bevor der Prozessor also die eigene lokale Simulationszeit auf den nächsten Eintrag in der Ereignisliste setzen kann, müßte sichergestellt sein, daß alle Ereignisse vor diesem Zeitpunkt abgearbeitet wurden.

Daraus folgt, daß eigentlich alle Ereignisse streng chronologisch abgearbeitet werden müßten. Damit wird deutlich, daß die ereignisgesteuerte Simulation eigentlich inhärent sequentiell ist. Wie man dennoch eine Parallelisierung erreichen kann, soll in den nächsten

Abschnitten gezeigt werden.

3 Auf der Suche nach der Parallelität

Bei der Verteilung von Simulationen gibt es zwei grundsätzliche Probleme. Das eine ist, wie teilt man das System auf mehrere Rechner auf, um eine größtmögliche parallele Abarbeitung zu erreichen. Im Idealfall sähe das so aus, daß man bei einer Verteilung der Simulation auf n Rechner eine Geschwindigkeitssteigerung um n im Gegensatz zu einem Rechner erhält. Es gibt Spezialfälle, in welchen dies erreicht werden kann. Im Normalfall jedoch fallen die Geschwindigkeitssteigerungen erheblich geringer aus. Das andere Problem bei der Verteilung ist die Kommunikation zwischen den Rechnern. Hierbei gibt es Fälle, in denen kaum Kommunikation notwendig ist, und andere, in denen sehr viel notwendig ist. Auch die Topologie des vorhandenen Netzwerkes ist dabei von Bedeutung.

In den folgenden Abschnitte sollen verschiedene Möglichkeiten der Verteilung von Simulationsaufgaben im Hinblick auf die Probleme der Parallelisierung und der Kommunikation dargestellt werden. Der erste Abschnitt bezieht sich dabei auf vorteilhafte Spezialfälle, während der zweite sich mit mit allgemeinen Strategien beschäftigt.

3.1 Superechner und Unabhängigkeiten

Die folgenden Spezialfälle nutzen Gegebenheiten des zu simulierenden Systems oder der Hardware aus, um die beiden Probleme der Parallelisierung und Kommunikation zu bewältigen.

3.1.1 Superechner Die einfachste Methode, Simulationen zu parallelisieren, ist, einen Parallelrechner mit entsprechendem parallelisierenden Compiler zu benutzen. Man kann dann eine sequentielle Simulation nehmen und hat keine Arbeit mit der Parallelisierung. Solche Compiler parallelisieren in der Regel aber nur im Kleinen, das heißt vereinfacht ausgedrückt, anstehende Prozessorbefehle werden auf ihre Parallelisierbarkeit überprüft und dann gleichzeitig ausgeführt. Unter Umständen erreicht man hiermit kaum eine parallele Ausführung, und vor allem wird die oft schon im Modell steckende Parallelität nicht ausgenutzt.

Superrechner sind aber immer dann von Vorteil, wenn es auf Echtzeitsimulationen ankommt, wie zum Beispiel bei Flugsimulatoren. Die Kommunikation zwischen den Systemteilen ist hier kein Problem, da auf gemeinsamen Speicher oder einen schnellen Datenbus zurückgegriffen werden kann.

3.1.2 Unabhängige Simulationsläufe Eine weitere Möglichkeit der einfachen Parallelisierung ist die Ausnutzung der Notwendigkeit unabhängiger Simulationsläufe. Um zum Beispiel stochastische Signifikanz zu erreichen oder um einfach mehrere Ergebnisse zu erhalten, müssen mehrere Simulationsläufe mit veränderten Parametern durchgeführt

werden. Bei Wettervorhersagen beispielsweise sollte man die Simulation mit unterschiedlichen Werten starten, um mit ausreichender Wahrscheinlichkeit Aussagen über den weiteren Verlauf des realen Wetters treffen zu können. Genauso sind bei Crashtests die Resultate für unterschiedliche Geschwindigkeiten und Aufprallwinkel interessant.

Somit kann man also auf n Rechnern n Simulationen starten, welche nicht einmal miteinander kommunizieren müssen während der Berechnungen. Dadurch ist ein theoretischer Speedup von n zu erreichen. Eine eventuell notwendige Kommunikation beschränkt sich auf das Starten der Simulationen mit den unterschiedlichen Startwerten und auf das Einsammeln der Ergebnisse. Damit ist auch die Topologie des benutzten Netzwerkes weniger von Bedeutung. Es bietet sich jedoch eine sternförmige Anordnung an mit einem zentralen Verteiler, der zudem eventuell benötigte Zwischenergebnisse einsammeln könnte. Für eine direkte parallele Verarbeitung der einzusammelnden Zwischenergebnisse würde sich auch eine Baumstruktur anbieten, bei ein Knoten die Daten der Söhne schon vorverarbeitet.

Auch wenn dieses Verfahren für alle Simulationen mit vielen unabhängigen Simulationsläufen am besten geeignet erscheint, kann es sinnvoller sein, besser einzelne Läufe zu beschleunigen. Dies ist zum Beispiel der Fall, wenn ein einziger Lauf einer Wettervorhersage schon 2 Tage dauern würde.

3.1.3 Funktionale Verteilung Eine Vorgehensweise zur parallelen Verarbeitung, die mehr Aufwand erfordert, ist die funktionale Verteilung. Bei Simulationen gibt es einige Aufgaben, die nichts direkt mit der Simulation selbst zu tun haben. Es müssen beispielsweise Pseudozufallszahlen berechnet, Steuerungsaufgaben erledigt und Ergebnisse eingesammelt werden. Derartige Aufgaben können gut auf die vorhandenen Prozessoren aufgeteilt werden. Hierbei fällt allerdings einige Kommunikation an, die bewältigt werden muß. Nach [MM89] bringt die funktionale Verteilung jedoch kaum nennenswerte Geschwindigkeitssteigerungen bei den meisten Simulationen, wenn man effiziente Algorithmen benutzt.

3.1.4 Simulation von Netzwerken Die Simulation von Netzwerken selbst wird sehr häufig benötigt. Hierbei möchte man das Design von weit verteilten Systemen und die Effizienz von Kommunikationsprotokollen untersuchen [Yan93]. Dazu bietet es sich natürlich an, die Knoten des zu simulierenden Netzwerkes auf lokale Rechner abzubilden. Die Rechner simulieren dann ein zu testendes Kommunikationsprotokoll und benutzen zur Kommunikation untereinander die üblichen Mechanismen. Zwischen diese beiden Protokolle muß außerdem noch ein Modul gesetzt werden, welches propagation delay, packet drops und Bandbreitenbegrenzung simuliert. Der Zeitverlust durch die Kommunikation zwischen den lokalen Rechner kann dabei direkt eingearbeitet werden, da er normalerweise erheblich geringer ist als bei der Kommunikation zwischen weit entfernten Rechnern.

3.2 Optimistische und pessimistische Strategien

Während die oben beschriebenen Verfahren spezielle Gegebenheiten ausgenutzt haben, soll hier beschrieben werden, wie man bei einem Großteil der ereignisgesteuerten Simulationen eine parallele Verarbeitung erreichen kann.

Die Verteilung wird hier so vorgenommen, wie im Abschnitt über ereignisgesteuerte Simulation dargestellt wurde. Die einzelnen Objekte werden also den vorhandenen Rechnern zugeteilt und kommunizieren dann über Nachrichten. Als Topologie des Netzwerkes wäre es deshalb am besten, wenn jeder Rechner mit jedem anderen verbunden ist. Da dies in der Regel jedoch nicht realisierbar ist, kann man versuchen, das System so aufzuteilen, daß die Teile, die am häufigsten Nachrichten austauschen, auch die schnellsten Verbindungen haben. Da normalerweise jedem Rechner mehrere Objekte zugeordnet werden, erhält jedes Objekt einen eigenen Prozeß. Die Simulation läuft also so ab, daß ein Prozeß ein Ereignis bearbeitet und sich aus diesem Ereignis Konsequenzen für andere Prozesse ergeben, die diesen über Nachrichten mitgeteilt werden. Dabei besitzt jeder Prozeß eine eigene Simulationszeit, wodurch überhaupt erst eine effiziente parallele Verarbeitung erreicht werden kann. Wenn ein Prozeß ein Ereignis bearbeitet hat, kann er nach Möglichkeit seine Simulationszeit erhöhen und so schon das nächste Ereignis bearbeiten.

An dieser Stelle sollen noch einmal die wichtigsten Merkmale des Beispiels Billard dargestellt werden: Die einzelnen Kugeln sind die eigenständigen Objekte, die den Prozessen zugeordnet werden. Die relevanten Ereignisse sind die Kollisionen zwischen den Kugeln oder mit dem Umrandung. Zur Verdeutlichung sei angenommen, daß auf der vorhandenen Hardware trigonometrische Berechnungen äußerst lange dauern. Die "komplizierten" Berechnungen also, die möglichst parallel ablaufen sollen, sind die neuen Bewegungsrichtungen und -geschwindigkeiten nach Kollisionen.

Es gibt nun grundsätzlich zwei verschiedene Strategien, mit welchen man das Problem der Parallelisierung angehen kann. Diese werden konservative oder pessimistische und optimistische Strategien genannt.

3.2.1 Die pessimistische Strategie Bei der pessimistischen Strategie erhöht ein Prozeß seine Simulationszeit immer nur dann auf das nächste Ereignis, wenn er sicher sein kann, daß kein anderes Ereignis dazwischenkommen kann. Um dies zu gewährleisten, erhalten alle Nachrichten, die verschickt werden, einen Zeitstempel mit der aktuellen Zeit des Senders. Weiterhin muß feststehen, welcher Prozeß welchem anderen eine Nachricht schicken kann. Das bedeutet, daß ein Prozeß bestimmte Eingänge und bestimmte Ausgänge besitzt. Wenn nun an allen Eingängen Nachrichten anliegen, kann der Prozeß seine Simulationszeit auf die Zeit der Nachricht mit dem kleinsten Zeitstempel erhöhen. Wenn allerdings auch nur ein Eingang ohne Nachricht vorhanden ist, muß der Prozeß warten, da hier eine Nachricht eintreffen könnte, die einen Zeitstempel vor allen anderen Nachrichten trägt. Das ganze sei an Bild 57 verdeutlicht. Prozeß P2 ist auf die dargestellte Art mit den Prozessen P1, P3 und P4 verbunden. Die Nachrichten werden durch Paare (t,m) dargestellt, wobei t der Zeitstempel und m der Inhalt sei. Da an dem Eingang von P3 her die Nachricht mit dem kleinsten Zeitstempel anliegt und kein Eingang ohne Nachricht ist, kann P2 nun seine Simulationszeit auf 2 erhöhen und die Nachricht

(5,m3) verschicken.

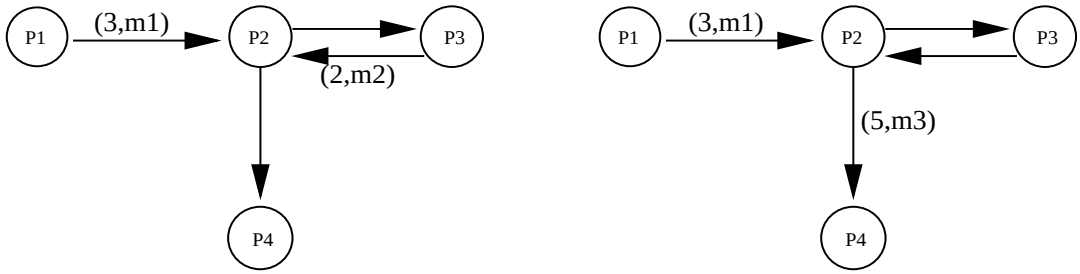

Abbildung 57. Eine Verklemmung liegt vor.

Bei dieser Strategie kann es nun sehr leicht zu Verklemmungen (deadlocks) kommen. In dem Beispiel war die einzige Konsequenz auf die Nachricht von P3 ein Nachricht zu P4. Somit liegt nun eine Verklemmung vor. P3 wartet auf eine Nachricht von P2 und umgekehrt. Eine Möglichkeit, solche Verklemmungen zu vermeiden, sind sogenannte Nullnachrichten. Dafür ist es erforderlich, daß jeder Prozeß eine untere Grenze für eine Zeitspanne hat, innerhalb welcher er alle Nachrichten voraussagen kann, welche er verschicken wird. Wenn er also weiß, daß bis zu einem bestimmten Zeitpunkt keine Nachrichten zu schicken sind, kann er eine Nullnachricht mit entsprechendem Zeitstempel schicken. Bild 58 verdeutlicht dies. Hier sind zusätzlich die lokalen Zeiten mit angegeben. P2 hat nun die lokale Zeit 2 und wartet auf P3. Da P2 z.B. zwei Zeiteinheiten vorausschauen kann, schickt er eine Nullnachricht an P3. P3 hat nun diese Nachricht als einzige Nachricht und kann somit seine Zeit auf 4 setzen. Danach schickt er ebenso eine Nullnachricht an P2. P2 hat jetzt wieder an allen Eingängen ein Nachricht anliegen und kann die Nachricht (3,m1) verarbeiten, was z.B. zu der Nachricht (8,m4) führt.

Eine weitere Möglichkeit, gegen Verklemmungen vorzugehen, ist, Verklemmungen festzustellen und sie dann zu beheben. Wenn also ein Prozeß eine Verklemmung feststellt, kann er bei dem betreffenden Vorgänger dessen lokale Zeit erfragen, um so weiterzukommen. Ist nun die lokale Zeit dieses Vorgängers größer als seine eigene, schickt der Vorgänger diese zurück. Ansonsten fragt dieser wiederum bei seinen Vorgängern nach.

3.2.2 Die optimistische Strategie Die optimistische Strategie geht davon aus, daß zunächst keine unerwarteten Ereignisse auftreten. Dies sei nochmals an dem Billardbeispiel erläutert. Der Prozeß für Kugel 8 behandele gerade das Ereignis, daß die Kugel in einem bestimmten Winkel und mit bestimmter Kraft von Kugel 3 angestoßen wurde. Nun überprüft er, ob auf der resultierenden Strecke ein anderes Hindernis liegt bzw. eine von den gerade rollenden Kugeln kreuzt. Angenommen, das nächste Hindernis sei der Umrandung, also plant der Prozeß die Kollision mit der Umrandung als nächstes Ereignis ein. Dies machte er, weil er optimistisch ausgelegt ist und nicht mit unerwarteten Ereignissen, d.h. Kollisionen, auf dem Weg zur Umrandung rechnet. Der Prozeß erhöht nun die Simulationszeit auf den Zeitpunkt der Kollision mit der Umrandung und berechnet das weitere Vorgehen.

Nun kann es natürlich vorkommen, daß er an der Umrandung angekommen feststellt,

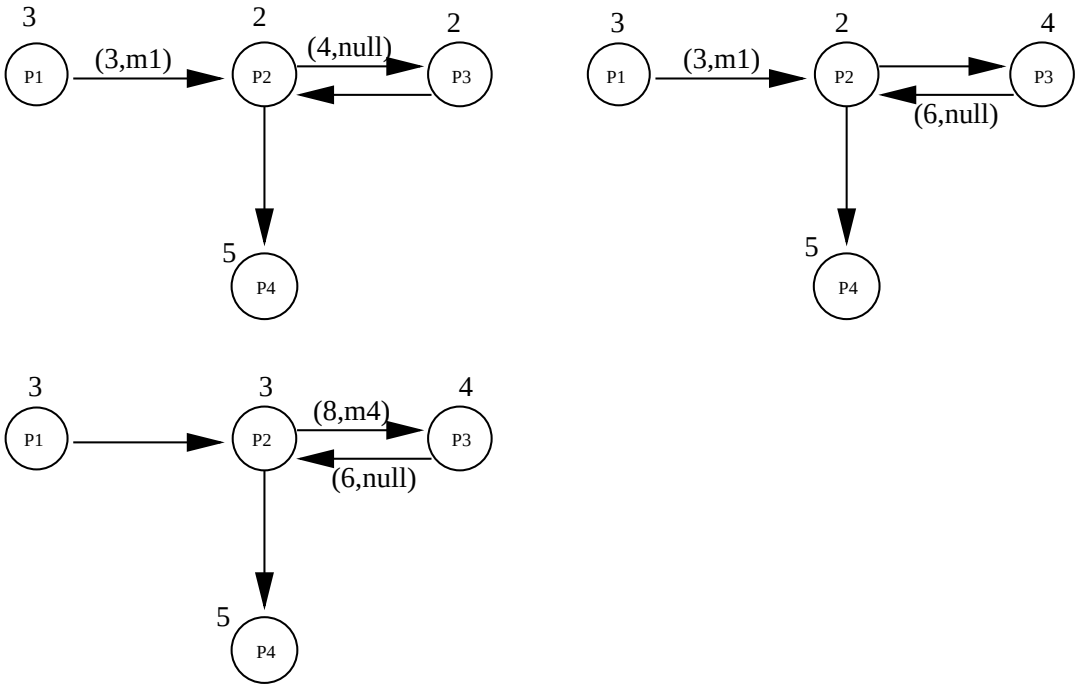

Abbildung 58. Auflösung der Verklemmung.

daß er eigentlich unterwegs mit Kugel 3 hätte kollidieren müssen, da diese nach ein paar Kollision mit der Umrandung wieder die Bahn kreuzte. In diesem Fall muß ein sogenannter "Roll-Back" durchgeführt werden. Das bedeutet, daß wieder der Zustand wiederhergestellt wird, der zum Zeitpunkt des betreffenden Ereignisses bestand.

Realisiert wird das ganze zum Beispiel durch den sogenannten Time Warp Algorithmus. Dieser funktioniert folgendermaßen: Wenn ein Prozeß eine Nachricht erhält, die eigentlich schon vor der aktuellen lokalen Zeit hätte bearbeitet werden müssen, setzt er seinen Zustand auf den Zeitpunkt des Zeitstempels der Nachricht zurück. Außerdem werden "Antinachrichten" geschickt, d.h. alle Nachrichten, die nach dem betreffenden Zeitpunkt geschickt wurden, werden zurückgerufen. Wenn nun ein Prozeß solch eine "Antinachricht" erhält, gibt es mehrere Möglichkeiten. Da beim Time Warp Algorithmus keine Reihenfolgetreue eingehalten werden muß, kann es sein, daß die ursprüngliche Nachricht, auf die sich die "Antinachricht" bezieht, noch gar nicht angekommen ist. Dann wird die "Antinachricht" einfach bis zur Ankunft gespeichert und hebt dann die positive Nachricht auf. Das gleiche passiert auch, wenn die positive Nachricht noch gar nicht bearbeitet wurde. Wenn sie schon bearbeitet wurde, muß dieser Prozeß ebenso einen "Roll-back" durchführen.

Diese Vorgehensweise, daß alle möglicherweise falschen Nachrichten zurückgenommen werden, nennt sich "aggressive cancellation". Es gibt eine "lazy cancellation". Hierbei werden zunächst keine "Antinachrichten" geschickt, sondern erst, wenn sich im weiteren Verlauf herausstellt, daß eine Nachricht eigentlich hätte anders laufen müssen.

Es müssen also bei diesem Algorithmus alle Zustände bis zur letzten sicher korrekten Zeit gespeichert werden. Diese korrekte Zeit ist das Minimum alle lokalen Zeiten.

3.3 Vergleichsdaten

Die Überschrift *Vergleichsdaten* mag etwas irreführend sein, da es immer sehr schwierig ist, verschiedene Messungen miteinander zu vergleichen. Um jedoch einen groben Überblick zu erhalten, in welcher Größenordnung sich die Geschwindigkeitssteigerungen bewegen, sollen hier ein paar Beispiele genannt werden. Der Speedup wird hierbei entweder als Speedup von x bei y Prozessoren oder in Prozent angegeben. Ersteres bedeutet die x-fach schnellere Ausführung bei der Benutzung von y Prozessoren statt einem. Die Prozentangabe setzt die beiden Größen in Relation zueinander. 100% bedeutet z.B. einen Speedup von x bei x Prozessoren.

[Lud87] beschreibt ein zeitgesteuertes, quasi-kontinuierliches Simulationsexperiment auf dem Erlanger 26-Prozessor-DIRMU-Experimentalsystem zur Lösung der Boltzmann-Gleichung aus dem Bereich der kinetischen Wärmetheorie. Der beste gemessene Speedup betrug hierbei 16 bei 26 Prozessoren. Ein anderes Beispiel beschreibt bei einem konservativen Ansatz mit Nullnachrichten auf einem System mit 17 Prozessoren und einer toroiden Topologie einen Speedup von 75%. Bei einem konservativen Ansatz mit dem Time Warp Algorithmus konnte teilweise ein Speedup von 90% erreicht werden.

4 Zusammenfassung

Es ist deutlich geworden, daß die Verteilung einer Simulation immer im Zusammenhang mit den gewünschten Ergebnissen und den vorhandenen Gegebenheiten betrachtet werden muß. Liegt einer der Spezialfälle vor, bietet es sich an, eine Vorgehensweise wie in Abschnitt 3.1 zu wählen. Wählt man dagegen eine Verteilung der einzelnen Objekte, ist zu überlegen, ob eine konservative oder eine optimistische Strategie zur Parallelverarbeitung benutzt werden soll. Der konservative Ansatz hat weniger Speicherbedarf als der optimistische, braucht aber eine festgelegte Verbindungshierarchie. Außerdem treten häufiger Totzeiten durch Warten auf. Insgesamt scheint der optimistische Ansatz mehr Potential zur Geschwindigkeitssteigerung zu bieten. Außerdem ist er flexibler, da die Verbindungen zwischen den Prozessen nicht festgelegt sein müssen wie beim konservativen Ansatz.

Interim Local Management Interface

Matthias Weng

Kurzfassung

Dieser Vortrag beschäftigt sich mit der vom ATM-Forum entwickelten Schnittstelle zum Management von privaten ATM-Netzwerken (ILMI). Die Management Information Base dieser Schnittstelle wird vorgestellt und mit einer Management Information Base der Internet Engineering Task Force verglichen.

1 Einleitung

Mit der Einführung von ATM-Netzwerken wurden auch neue Anforderungen an das Netzwerkmanagement gestellt. Es galt nun die verschiedenen Verkehrstypen, den Durchsatz und die ganze Bandbreite von Bedingungen und Optionen, die ATM bereithält, zu verwalten. Da es auf diesem Gebiet bisher noch keine einheitliche und umfassende Lösung gab, entwickelte das ATM-Forum eine Übergangsversion für ein Management Interface, das sogenannte ILMI (Interim Local Management Interface)[AF95].

Im folgenden soll zuerst das Management von ATM-Netzwerken betrachtet werden, bevor die Eigenschaften der ILMI und ihrer Management Information Base vorgestellt werden. In Kapitel 4 wird das Interface des ATM-Forums mit dem der Internet Engineering Task Force verglichen.

2 Management von ATM-Netzen

2.1 ATM-Netze

Durch den größer werdenden Bedarf an Netzkapazität gewinnen hohe Übertragungsgeschwindigkeiten, wie sie ATM ermöglicht, immer mehr an Bedeutung. ATM Verbindungen erlauben Übertragungsraten von 155 Mbit/s (in Zukunft sogar noch mehr). Durch Paketvermittlung und Zeitmultiplex ist es jedoch auch möglich, mit verschiedenen Geschwindigkeiten zu übertragen, die vor der Übertragung mit der Empfängerseite „ausgehandelt“ werden. Vor dem Versenden wird eine Nachricht von der ATM - Adaptionsschicht (ATM Adaptation Layer, AAL) in Pakete zerlegt und abwechselnd mit anderen Nachrichtenpaketen versandt. Im Zellkopf ist dabei eine Verbindungskennung enthalten, so daß jedes Paket über mehrere ATM-Vermittlungsstellen (Switches) zu seinem Bestimmungsort gelangt, wo es mittels der ebenfalls im Zellkopf enthaltenen Informationen mit den zugehörigen Paketen in der richtigen Reihenfolge wieder zusammengesetzt wird.

Eine Ende-zu-Ende Verbindung wird mit virtuellen Kanälen hergestellt, von denen jeder Kanal durch eine Nummer gekennzeichnet ist (Virtual Channel Identifier, VCI). Mehrere virtuelle Kanäle können zu einem Virtuellen Pfad zusammengesetzt sein, der ebenfalls einen Identifikator besitzt (Virtual Path Identifier, VPI). Die Kombination von VPI und

VCI stellt die erwähnte Verbindungskennung dar, die bei ATM nur lokale Bedeutung hat, da sie in allen Switches umgesetzt werden kann.

2.2 Netzwerkmanagement

Um das Funktionieren eines Netzes zu gewährleisten, bedarf es einer zentralen Instanz, dem Netzwerkmanager, der den Zustand des Netzes überwacht. Er kommuniziert über ein Managementprotokoll mit den Agenten, die an bestimmten Geräten wie z.B. Gateways, Routern oder Workstations plaziert sind (s. Abb. 59). Als ein Standard hat sich dabei das Simple Network Management Protocol (SNMP) des Internets durchgesetzt.

Der Manager schickt eine Anfrage oder einen Befehl an einen Agenten, der daraufhin die gewünschte Information zurückgibt oder auf den Befehl reagiert. Ebenso kann der Agent bei auftretenden Fehlern eine Benachrichtigung (Trap) an den Manager schicken.

Es ist erforderlich, daß Manager und Agent auf einer gemeinsamen Datenbasis arbeiten, um sich gegenseitig zu „verstehen". Diese Datenbasis wird als Management Information Base (MIB) bezeichnet. In der MIB werden z.B. Vereinbarungen über Übertragungsarten und -medien, über bestehende virtuelle Verbindungen und Statistiken über das Netz festgehalten. Diese Objekte werden in ASN.1 (Abstract Syntax Notation No.1) formuliert.

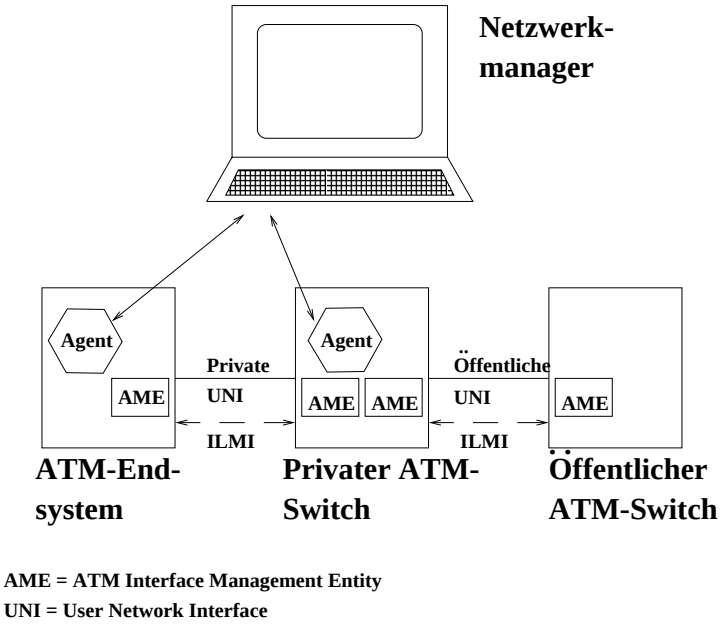

Abbildung 59. Manager und Agent

2.3 Das Managementmodell des ATM-Forums

Die vielen Anforderungen an einen Netzmanager veranlaßten das ATM-Forum, ein Managementmodell zu entwickeln, das die verschiedenen Managementpfade in ATM-Netzwerken definiert (s. Abb. 60). Es wurden fünf grundlegende Managementschnittstellen fest-

gelegt, die mit M1 bis M5 bezeichnet werden. M1 und M2 definieren jeweils die Schnittstellen zwischen der lokalen Managementeinheit und einem ATM-Endgerät beziehungsweise dem lokalen Netzwerk. Mit M3 wird die Schnittstelle zwischen den lokalen und den Managementsystemen des öffentlichen Netzbetreibers bezeichnet. M4 ist die Schnittstelle zwischen der Managementeinheit des Netzanbieters und dem öffentlichen Netz. M5 legt die Verbindung zwischen zwei Managementsystemen des Betreibers fest. Sie ist die komplexeste aller Schnittstellen und bis heute noch nicht näher spezifiziert.

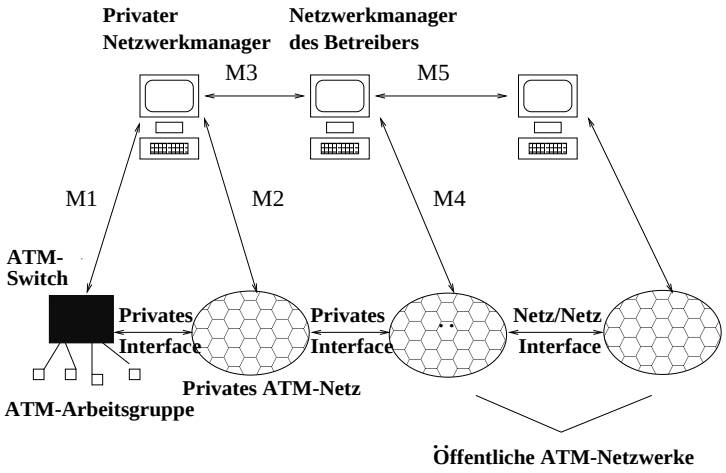

Abbildung 60. Rahmenwerk für ATM-Management des ATM-Forums

3 Funktionsweise der ILMI

In jedem ATM-Gerät ist eine Schnittstellen-Managementeinheit (ATM Interface Management Entity, AME) vorhanden, die die ILMI - Funktionen der Schnittstelle unterstützt. Ein Nachrichtenaustausch findet immer nur zwischen zwei benachbarten AMEs statt, die zu diesem Zweck auch auf die MIB-Daten der jeweils anderen AME zugreifen können (s. Abb. 59). Für die Kommunikation zwischen zwei AMEs wird ein spezieller virtueller Kanal reserviert. Über diesen Kanal werden Anfragen, Antworten und Traps als AAL-gekapselte SNMP-Nachrichten geschickt. Es gibt eine Voreinstellung für den ILMI-Kanal (VPI=0,VCI=16), die jedoch auch verändert werden kann. Die Cell Loss Priority für die Managementzellen ist 0, d. h., die Zellen dürfen auch bei zu hohem Datenaufkommen nicht verworfen werden. Als Richtlinie gilt, daß der SNMP-Verkehr nicht mehr als ein Prozent der Bandbreite einer Verbindung betragen sollte.

Alle Eigenschaften einer Schnittstelle sind als sogenannte Managed Objects in einer MIB festgehalten. Managed Objects stellen in diesem Fall Variablen dar, die gesetzt und ausgelesen werden können. Die MIB ist in Form eines Baumes organisiert, in dessen Blättern die Informationen enthalten sind. Die ISO hat einen Namensbaum für Managed Objects definiert, in den jede Organisation ihren eigenen Teilbaum „einhängen“ kann (s. auch [Sei94]). Die ILMI-MIB gliedert sich in den Namensbaum unter dem Knoten enterprises(353) ein (s. Abb. 61).

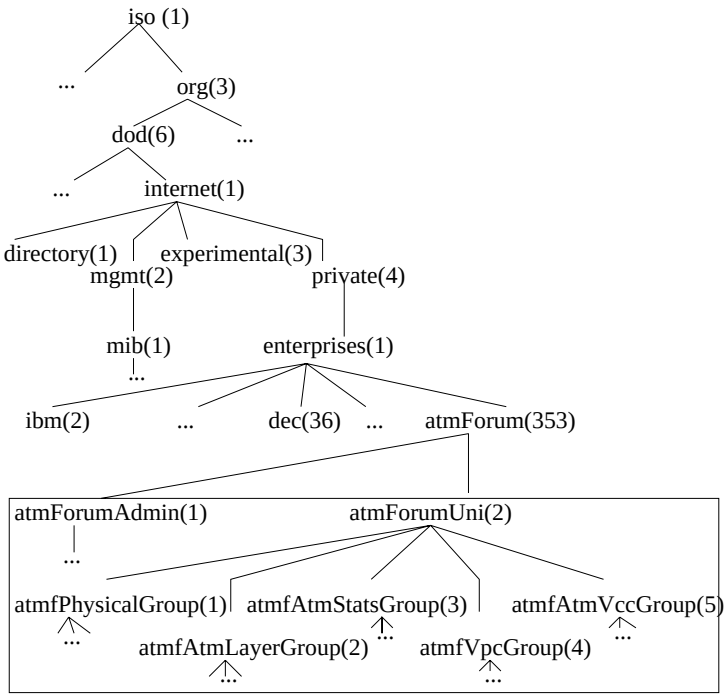

Abbildung 61. Der Namensbaum für Managed Objects

4 Die ILMI Management Information Base

Die ILMI - MIB unterteilt sich in zwei Unterbäume. Der Teilbaum *atmForumAdmin* enthält die Definitionen für mögliche Werte der Schnitttstellenobjekte, sogenannte Object Identifier Definitions, und der Teilbaum *atmForumUni* die Schnittstellenobjekte selbst, die weiter in die MIB-Groups unterteilt sind.

4.1 Object Identifier Definitions

4.1.1 atmfTransmissionTypes In dieser Gruppe werden die neun verschiedenen Übertragungstypen definiert, die von der MIB zur Zeit unterstützt werden (unbekannt, Sonet STS-3c, DS3, 4B5B, etc.).

4.1.2 atmfMediaTypes Hier werden sechs verschiedene Medientypen vereinbart, die von der MIB unterstützt werden (unbekannt, Koaxialkabel, Single Mode Fiber, Multi Mode Fiber, Shielded Twisted Pair, Unshielded Twisted Pair).

4.1.3 atmfTrafficDescrTypes Acht verschiedene Verkehrsvereinbarungen werden derzeit von der MIB unterstützt. Diese umfassen Parameter wie Cell Loss Priority (CLP), Sustainable Cell Rate (SCR) und Tagging.

4.2 MIB Groups

4.2.1 Managed Objects der atmfPhysicalGroup Diese Gruppe beschreibt die Eigenschaften der physikalischen Verbindung. Sie besteht aus:

- der Tabelle *atmfPortTable* mit dem Unterknoten *atmfPortEntry* für die folgenden Einträge:
  - *atmfPortIndex*: Identifikation einer bestimmten physikalischen Schnittstelle eines ATM-Geräts, über einen Index.
  - *atmfPortAddress*: Dieses Objekt enthält die ATM-Adresse dieses Ports. In der neuen Version der UNI ist eine separate MIB zur Registrierung von Adressen vorgesehen. Aus diesem Grund sollte es nicht mehr implementiert werden, außer wenn ein Abwärtskompatibilität zu früheren UNIs erforderlich sein sollte.
  - *atmfPortTransmissionType*: Dieses Objekt gibt an welche der in *atmfTransmissionTypes* vereinbarten Übertragungsarten (s. 4.1.1) fuer diese physikalische Schnittstelle gewählt wurde(z.B. *atmfSonetSTS3c* für Sonet STS-3c mit 155.52 Mbit/s).
  - *atmfPortMediaType*: Hier ist festgelegt welches der in *atmfMediaTypes* vereinbarten Übertragungsmedien (s. 4.1.2) an dieser Schnittstelle verwendet wird. (z.B. *atmfMediaCoaxCable* für ein Koaxialkabel).
  - *atmfPortOperStatus*: Der Status zeigt an, ob die physikalische Schnittstelle aktiv (In-Service), nicht aktiv (Out-of-Service) oder im Loop Back Mode (Loop Back) ist.
  - *atmfPortSpecific* : Dieses Objekt enthält zusätzliche spezifische Übertraguns- und Medieninformationen.
  - *atmfPortMyIfName*: Hier wird der Name dieser Schnittstelle festgehalten.
- *atmfMyIpNmAddress* und *atmfMyOsiNmNsapAddress*: Im Normalfall wird von einer AME nur eines dieser beiden Objekte unterstützt. Sie enthalten jeweils die IP-oder die NSAP-Adresse, die vom Netzwerkmanager zum Management des ATM-Geräts benutzt werden kann.
- *atmfMySystemIdentifier*: Ein 48 Bit Identifikator für das System.

4.2.2 Managed Objects der atmfAtmLayerGroup Diese Gruppe enthält Informationen über die ATM-Schicht. Wie die *atmfPhysicalGroup* besteht sie aus einer Tabelle, *atmfAtmLayerTable* bezeichnet, mit dem Unterknoten *atmfAtmLayerEntry* für die Einträge:

- *atmfAtmLayerIndex*: Der Index zur Kennzeichnung der ATM-Schnittstelle.
- *atmfAtmMaxVPCs*: Hier wird festgelegt wieviele virtuelle Pfade maximal gleichzeitig unterstützt werden.
- *atmfAtmMaxVCCs*: Ähnlich wie im vorherigen Objekt wird hier angegeben wieviele virtuelle Kanäle maximal gleichzeitig unterstützt werden.
- *atmfAtmLayerConfiguredVPCs*: Hier wird die momentane Anzahl von virtuellen Pfaden festgehalten.

- *atmfAtmLayerConfiguredVCCs*: Dieses Objekt hält die momentane Anzahl von Virtuellen Kanälen fest.
- *atmfAtmLayerMaxVpiBits*: An dieser Stelle ist angegeben wieviel Bits maximal für die VPIs zur Verfügung stehen.
- *atmfAtmLayerMaxVciBits*: Dieses Objekt legt fest wieviel Bits maximal für die VCIs zur Verfügung stehen.
- *atmfAtmLayerUniType*: Hier ist der ATM-Schnittstellentyp angegeben. Er kann vom Typ „public“ (öffentlich) oder „private“ (privat) sein.
- *atmfAtmLayerUniVersion*: Hier erfolgt die Angabe der neuesten ATM-Forum UNI Spezifikation, die von dieser Schnittstelle unterstützt wird.
- *atmfAtmLayerDeviceType*: Dieses Objekt bestimmt die Rolle der referenzierten Schnittstelle „user“ (Benutzer) oder „node“ (Knoten).

4.2.3 Managed Objects der atmfAtmStatsGroup Diese Gruppe muß nicht implementiert sein. Sie enthält eine Statistik über die empfangen und gesendeten ATM-Zellen. Wie die vorigen besteht auch sie aus einer Tabelle, dem Knoten *atmfAtmStatsTable*, deren Einträge im Unterknoten *atmfAtmStatsEntry* stehen.

- *atmfAtmStatsIndex*: Es wird wiederum die Nummer der ATM-Schnittstelle als Index verwendet.
- *atmfAtmStatsReceivedCells*: Hier wird die Anzahl der erhaltenen ATM - Zellen an diesem Interface, die nicht verworfen wurden festgehalten.
- *atmfAtmStatsDroppedReceivedCells*: In diesem Objekt ist die Anzahl der verworfenen ATM - Zellen enthalten. Zellen können verworfen werden aufgrund von
  - einer Fehlermeldung der Zellkopf - Fehlerkontrolle
  - falschem Zellkopfformat
  - falschem VPI oder VCI.
- *atmfAtmStatsTransmittedCells*: Dieses Objekt enthält die Anzahl der gesendeten ATM - Zellen.

4.2.4 Managed Objects der atmfVpcGroup In dieser Gruppe zur Verwaltung der virtuellen Pfade (Virtual Path Connection, Vpc) sind nur permanente Verbindungen repräsentiert. Auch sie besteht aus einer Tabelle mit der Bezeichnung *atmfVpcTable*, deren Einträge im Unterknoten *atmfVpcEntry* stehen.

- *atmfVpcPortIndex*: Es wird der gleiche Interface Index wie in den vorherigen Gruppen verwendet.
- *atmfVpcVpi*: Hier ist der VPI für den aktuellen Pfad festgehalten.

- *atmfVpcOperStatus*: Zur Kennzeichnung des Arbeitsstatus sind die Werte „unknown“, „end2endUp“, „end2endDown“, „localUpEnd2endUnknown“ und „localDown“ vorgesehen..

- *atmfVpcTransmitTrafficDescriptorType* und *atmVpcTransmitTrafficDescriptorParam1-5*: Je nach angegebenem Typ sind 0 bis 5 Parameter vorhanden, die jeweils die Eigenschaften des ausgehenden Verkehrs beschreiben. Die Parameter können dabei die Werte annehmen, die in der Gruppe *atmfTrafficDescrType* (s. 4.1.3) vereinbart sind (z.B. *atmfClpTaggingScr*).

- *atmfVpcReceiveTrafficDescriptorType* und *atmfVpcReceiveTrafficDescriptorParam1-5*: Wie im vorherigen Objekt sind je nach angegebenem Typ 0 bis 5 Parameter vorhanden, die jeweils die Eigenschaften des Eingangverkehrs beschreiben. Auch hier sind nur die vereinbarten Typen zulässig.

- *atmfVpcQoSCategory*: Dieses Objekt sollte nicht mehr implementiert werden, außer wenn eine Abwärtskompatibilität mit der Version 2.0 der UNI benötigt wird.

- *atmfVpcTransmitQoSClass*: Hier wird die Übertragungsqualität beim Senden festgelegt.

- *atmfVpcReceiveQoSClass*: Analog zum vorherigen Objekt erfolgt hier die Festlegung der Übertragungsqualität beim Empfang.

4.2.5 Managed Objects der atmfVccGroup Diese Gruppe zur Verwaltung der virtuellen Kanale (Virtual Channel Connection, Vcc) gleicht der *atmfVpcGroup*. Die Tabelle atmfVccTable enthält wieder den Unterknoten atmfVccEntry, der die folgenden Objekte enthält: *atmfVccPortIndex*, *atmfVccVpi*, *atmfVccOperStatus*, *atmfVccTransmitTrafficDescriptorType*, *atmVccTransmitTrafficDescriptorParam1-5*, *atmfVccReceiveTrafficDescriptorType*, *atmfVccReceiveTrafficDescriptorParam1-5*, *atmfVccQoSCategory*, *atmfVccTransmitQoSClass* und *atmfVccReceiveQoSClass*. Die Funktion eines Objekts entspricht jeweils der Funktion des äqiuvalenten Objekts in Abschnitt 4.2.4. Zusätzlich gibt es hier noch das Objekt *atmfVccVci*, das den VCI beschreibt.

4.2.6 Traps Für die ILMI wurden zwei Traps definiert. Es gibt den Trap *atmfVccChange* der das Löschen oder Einrichten eines virtuellen Kanals anzeigt und den Trap *atmfVpcChange* für den selben Vorgang bei virtuellen Pfaden.

5 Vergleich der ILMI - MIB mit der ATM - MIB der IETF (RFC 1695)

Außer der vom ATM-Forum entwickelten ILMI-MIB gibt es auch noch eine Reihe weiterer MIBs für ATM-Netze, die von anderen Arbeitsgruppen erstellt wurden, so z.B. die AToM-MIB der Internet Engineering Task Force (IETF) [AT94]. Die Gruppe stützte sich

dabei auf eine Menge schon vorhandener MIBs, wie die ILMI-MIBs, die Sonet MIB und andere. Die AToM - MIB konzentriert sich derzeit hauptsächlich auf das Management von permanenten virtuellen Verbindungen (PVCs) und nicht auf geschaltete virtuelle Verbindungen (SVCs).

Im Gegensatz zur ILMI-MIB gibt es bei der AToM-MIB allerdings einen ausführlichen Teil zur Verwaltung der Weitervermittlung von Verbindungen wie sie hauptsächlich in Knoten anzutreffen ist. Ein weiterer Schwerpunkt der AToM-MIB liegt auf dem Management von AAL5 Verbindungen für die es in der MIB eine eigene Gruppe gibt. Die AToM-MIB ist ebenfalls im ISO-Namensbaum eingegliedert, und zwar unter dem Knoten mib (37) (s. Abb. 60).
Die Managed Objects der AToM MIB sind im wesentlichen in den folgenden Gruppen eingeordnet :

- *atmInterfaceConfTable*: Diese Gruppe entspricht der *atmfPhysicalGroup* bei der ILMI-MIB. Wie dort sind auch hier die Informationen über die aktive Anzahl von VCCs und VPCs, über die maximale Zahl von VCIs und VPIs und über die Schnittstelle enthalten. Zusätzlich gibt es noch die zwei Objekte *atmInterfaceIlmiVpi* und *atmInterfaceIlmivci*, die angeben, welcher Pfad und welcher Kanal an dieser ATM-Schnittstelle die ILMI unterstützen.

- *atmTrafficDescrParamTable*): Im Gegensatz zur ILMI-MIB, wo die Beschreibung der Verkehrsparameter in den VCC und VPC-Gruppen erfolgt werden diese Parameter hier zentral vereinbart und die anderen Gruppen greifen darauf zu.

- *atmVplTable*:Diese Funktion Gruppe entspricht der *atmfVpcGroup* (Virtuelle Pfade) in ihrer Funktionalität. Zusätzlich ist noch ein Identifikator *atmVplCrossConnectIdentifier* vorhanden, wenn es sich um eine weitervermittelte Verbindung handelt.

- *atmVclTable*: Diese Gruppe entspricht der *atmfVccGroup* bei der ILMI. Wie beim vorherigen Objekt ist ein Identifikator *atmVclCrossConnectIdentifier* für weitervermittelte Verbindungen vorhanden.

- *atmInterfaceDs3PlcpTable*: Eine spezielle Gruppe für Schnittstellen, die DS3 PLCP als Übertragungsart benutzen.

- *atmInterfaceTCTable*: Eine spezielle Gruppe für Schnittstellen, die TC Sublayer als Teilschicht zur Anpassung an die Übertragungsart verwenden.

- *atmVpCrossConnectTable*: In dieser Gruppe werden die weitervermittelten Verbindungen (Cross-connect) eingetragen. Es gibt einen ausführlichen Teil zum Aufbau und zur Verwaltung dieser Verbindungen. Jede Verbindung hat einen eigenen *atmVpCrossConnectIndex* der mit dem *atmVplCrossConnectIdentifier* aus der Gruppe *atmVplTable* übereinstimmt. Außerdem sind Informationen über Eingangs- und Ausgangs-VPI vorhanden.

- *atmVcCrossConnectTable*: Dies ist die zur vorhergehenden äquivalente Gruppe für virtuelle Kanäle. In dieser Gruppe werden nicht nur die weitervermittelten virtu-

ellen Pfade, sondern auch die virtuellen Kanäle verwaltet. Wie bei *atmVpCrossConnectTable* gibt es einen ausführlichen Teil zum Aufbau und zur Verwaltung dieser Verbindungen, bei denen hier über den eigenen *atmVcCrossConnectIndex* und über Eingangs- und Ausgangs-VCI und-VPI auf den einzelnen Kanal zugegriffen werden kann.

– *aal5VccTable*:Hier sind statistische Informationen über alle AAL5 Verbindungen, die in diesem Gerät enden, eingetragen. Dies ähnelt der ATM-Schicht Statistik in der Gruppe *atmfAtmLayerStats* der ILMI.

6 Zusammenfassung und Ausblick

In diesem Beitrag wurde die ILMI-Schnittstelle für das Management von privaten ATM-Netzen vorgestellt. In jeder AME, die die ILMI unterstützt ist die ILMI-MIB wie sie in Kapitel 4 beschrieben wurde vorhanden.

Die ILMI ist, ebenso wie auch die AToM-MIB, darauf beschränkt, die Schnittstelle zwischen zwei benachbarten AMEs zu managen. Sie ist nicht in der Lage Managementinformationen durchs Netz zu verteilen. Es ist allerdings zu erwarten, daß zukünftige Schnittstellenmanager dazu in der Lage sein werden. Die ILMI wurde nur als Übergangslösung entwickelt, aber man kann davon ausgehen, daß weitere Entwicklungen darauf aufbauen werden. Die ILMI ist somit als eine Art Vorschlag zu verstehen, der von den Anbietern von Managementanwendungen nach ihren Vorstellungen modifiziert werden kann, und der einen Einstieg in das Management von ATM-Netzen bietet.

Es wird mit Sicherheit noch einige Zeit dauern, bis es eine komplette Lösung für das Management von ATM - Netzen geben wird, und es gibt Bemühungen, die zukünftigen Operationen ohne SNMP durchzuführen. Dabei sollen intelligente Zellen, sogenannte OAM-Zellen (Operations, Admimistration and Maintenance) zum Einsatz kommen, die wie normale ATM - Zellen durchs Netz gehen und durch ihren Header als Managementzellen erkannt werden können. Diese Entwicklung würde dazu führen, daß erheblich weniger Managementverkehr im Netz herrschen würde und die Notwendigkeit zur Verteilung von MIBs zurückgehen würde. In Zukunft werden noch weitere OAM-Zellen definiert werden, doch es wird noch einige Jahre dauern, bis es eine Managementanwendung gibt, die auf diese Weise funktioniert.

Es bleibt auch noch weiterhin Spielraum um z.B.Schnittstellen zwischen ATM-Netzen und herkömmlichen Netzen zu entwickeln. Die Zielsetzung bei der Entwicklung neuer Systeme ist jedoch und sollte es auch bleiben, auf einer gemeinsamen, einheitlich definierten Basis (wie sie z.B. auch die ILMI darstellt) die nötigen anwender- und herstellerspezifische Ausprägungen anzubieten.

Fehler- und Leistungsmanagement in ATM-Netzen

Sven Buth

Kurzfassung

Der Beitrag stellt im wesentlichen die Vorschläge für das Fehler- und Leistungsmanagement von ATM-Netzen mit Hilfe von OAM-Zellen vor. Hierzu zählen zum Beispiel Loopback-Tests und Kontinuitätsüberprüfungen. Außerdem wird der Vorschlag für eine Management Information Base (MIB), die Loopback-Tests in ATM-Netzen ermöglicht, beschrieben. Es werden zudem einleitend die wichtigsten Begriffe aus der Welt des ATM-Netzwerkmanagements erklärt.

1 Einleitung

In großen Netzwerken gewinnt das Management zunehmend an Bedeutung. Insbesondere in modernen Hochleistungsnetzwerken werden spezielle Funktionen benötigt, die den Netzwerkadministrator bei dieser Aufgabe unterstützen. Das ATM-Forum hat hierfür bereits beim Entwurf des ATM-Standards ein Konzept eingeplant, das auf sogenannten OAM-Zellen (Operations and Maintenance) basiert. Dieser Beitrag zeigt die Möglichkeiten des Fehler- und Leistungsmanagements mit Hilfe dieser OAM-Zellen auf.

Im folgenden Kapitel wird zunächst kurz die grundlegende Architektur des ATM-Netzwerkmanagements beschrieben, insbesondere werden die in dieser Arbeit häufig verwendeten Begriffe Virtual Path (VP) und Virtual Channel (VC) erläutert.

Im dritten Kapitel wird zuerst der allgemeine Aufbau von OAM-Zellen vorgestellt, um dann auf die speziellen Eigenschaften in Bezug auf die einzelnen Funktionen des Fehler- und Leistungsmanagements in ATM-Netzen einzugehen.

Hieran schließt sich im vierten Kapitel die Untersuchung eines Vorschlags für eine Management Information Base (MIB) für Loopback-Tests in ATM-Netzen und deren Einordnung in das OAM-Konzept an.

2 ATM-Managementarchitektur

2.1 Die ATM-Schicht

Abbildung 62 zeigt, daß die ATM-Schicht direkt oberhalb der Physikalischen Schicht (SONET bzw. SDH) liegt. Die ATM-Schicht verwaltet die virtuellen Pfade (VP) und die virtuellen Kanäle (VC). Abbildung 63 zeigt den Zusammenhang zwischen virtuellen Pfaden und virtuellen Kanälen. Ein virtueller Pfad ist Teil einer physikalischen Verbindung und eindeutig durch den Virtual Path Identifier (VPI) gekennzeichnet. In einem virtuellen Pfad sind wiederum mehrere virtuelle Kanäle enthalten, die ebenfalls eindeutig (bezogen auf den Pfad) durch einen Virtual Channel Identifier (VCI) gekennzeinet sind.

Eine Verbindung von Instanzen oberhalb der ATM-Schicht wird als Virtual Channel Connection (VCC) bezeichnet. Diese setzt sich aus mehreren VCs zusammen. Die Verbindung zwischen den Endpunkten eines VCs wird als Virtual Path Connection (VPC) bezeichnet und besteht aus mehreren VPs.

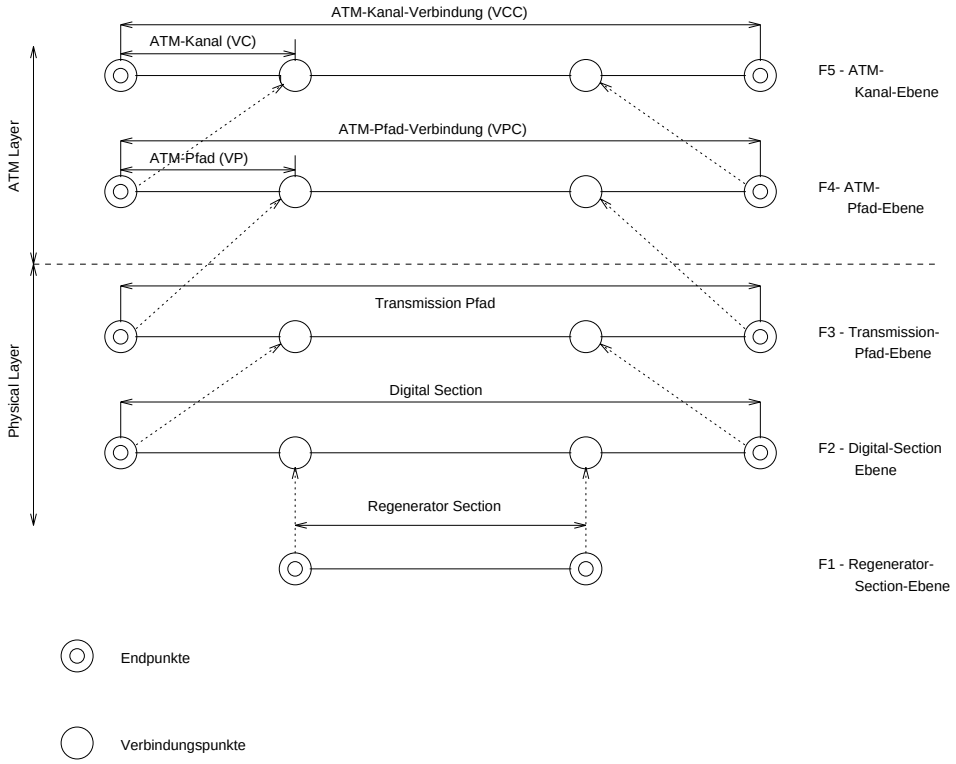

Abbildung 62. Informationsflüsse in den verschiedenen Protokollebenen

Abbildung 63. ATM-Kanäle (VC) und ATM-Pfade (VP)

Abbildung 64 zeigt diese Zusammenhänge an einem Beispiel.

2.2 Management

Das Management von ATM-Netzen erfolgt mit Hilfe von OAM-Zellen (Operations and Maintenance), die von den entsprechenden Knoten auf den gleichen ATM-Pfaden bzw. ATM-Kanälen verschickt werden wie die Daten. Im folgenden Kapitel wird der Aufbau und die Funktionsweise von OAM-Zellen näher erläutert.

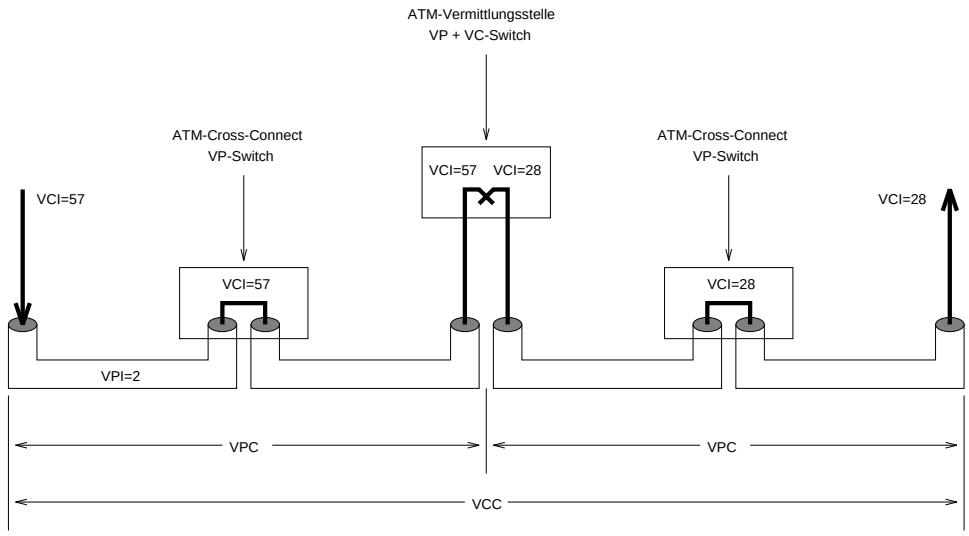

Abbildung 64. Darstellung einer virtuellen Verbindung

3 Management mit Hilfe von OAM-Zellen

3.1 Typen von OAM-Zellen

Wie in Abbildung 65 zu sehen ist, ist das Format für VPC (F4) OAM-Zellen und für VCC (F5) OAM-Zellen im Grunde das gleiche. Unterschiedlich ist lediglich die Bedeutung der einzelnen Felder. Während bei F4-Zellen der VCI angibt, ob es sich um eine Segment-Verbindung (VCI=3) oder um eine Ende-zu-Ende-Verbindung (VCI=4) handelt, wird bei F5-Zellen der VCI zur Identifikation der Verbindung benötigt. Daher wird hier über den Payload Type angeben, ob es sich um eine Segment-Verbindung (PT = 100) oder um eine Ende-zu-Ende-Verbindung (PT = 101) handelt.

Tabelle 2 zeigt die verschiedenen Arten von OAM-Zellen und die dazugehörigen Funktionstypen. Die angegebenen Kodierungen kennzeichnen die entsprechenden OAM-Zellen in den jeweiligen Feldern. Momentan sind OAM-Zellen für das Fehlermanagement, für das Leistungsmanagement und zur Aktivierung/Deaktivierung von bestimmten Funktionen definiert. Für jede Art existieren spezielle Funktionen. Diese umfassen bei Fehlermanagement-Zellen das Alarm Indication Signal (AIS), die Remote Defect Indication (RDI, Far End Reporting Failure FERF bedeutet das gleiche), die Continuity Check Funktion und die Loopback Funktion. Für Leistungsmanagement-Zellen sind die Funktionen Forward Monitoring und Backward Reporting vorgesehen. Außerdem gibt es noch Funktionen zum Aktivieren/Deaktivieren von Kontinuitätsüberprüfung und Performance Monitoring.

Die im folgenden beschriebenen Definitionen der funktionsspezifischen Felder der OAM-Zellen basieren auf der ATM Forum B-ICI Spezifikation [AF93]. Hierzu wird im Abschnitt 3.2 zuerst auf das Fehlermanagement eingegangen und in Abschnitt 3.3 dann auf das Leistungsmanagement.

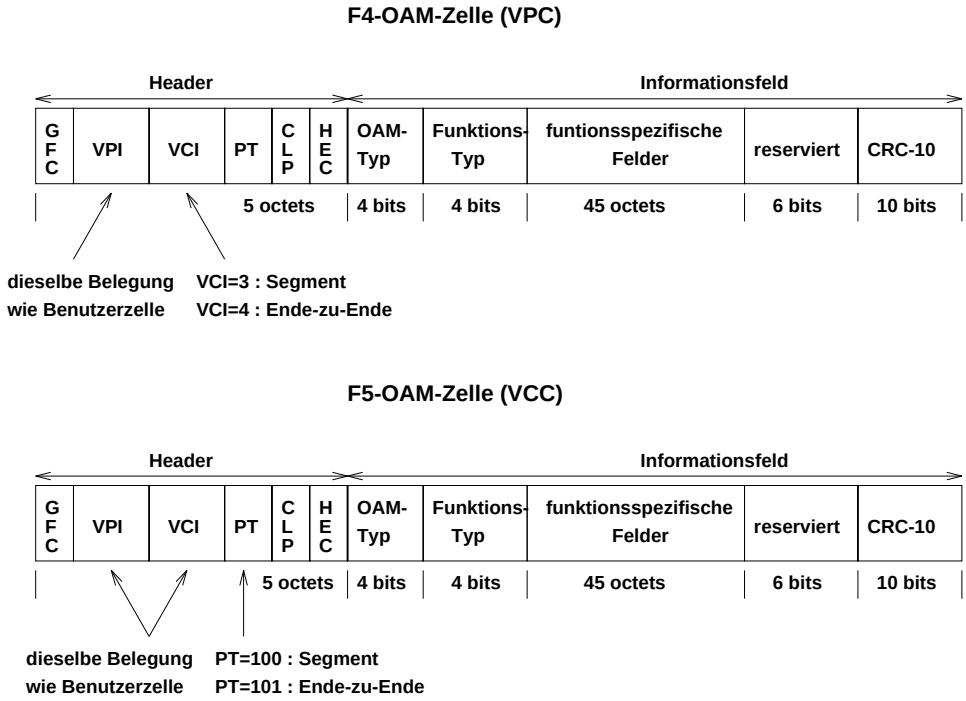

Abbildung 65. ATM OAM-Zellen Format

OAM Type		Function Type	
Fault Management	0001	AIS	0000
		RDI/FERF	0001
		Kontinuitätsüberprüfung	0100
		Loopback	1000
Leistungs-Management	0010	Forward Monitoring	0000
		Backward Reporting	0001
		Monitoring & Reporting	0010
Aktivierung/Deaktivierung	1000	Performance Reporting	0000
		Kontinuitätsüberprüfung	0001

Tabelle 2. OAM Typen und deren Funktionstypen

3.2 Fehlermangament

Das Fehlermanagement dient dazu, Fehler zu erkennen und andere, an der Verbindung beteiligte Elemente darüber zu informieren. Dafür vorgesehen sind die OAM-Zellen-Funktionstypen AIS, RDI/FERF, Kontinuitätsüberprüfung und Loopback-Test.

3.2.1 AIS und RDI/FERF Abbildung 66 zeigt die funktionsspezifischen Felder von AIS und RDI/FERF OAM-Zellen. Die einzelnen Felder haben folgende Bedeutung:

- *Failure Type* : Er zeigt an, welche Art von Fehler aufgetreten ist. Es sind noch keine speziellen Werte standardisiert.

- *Failure Location* : Sie gibt an, wo der Fehler aufgetreten ist. Es sind noch keine speziellen Werte standardisiert.

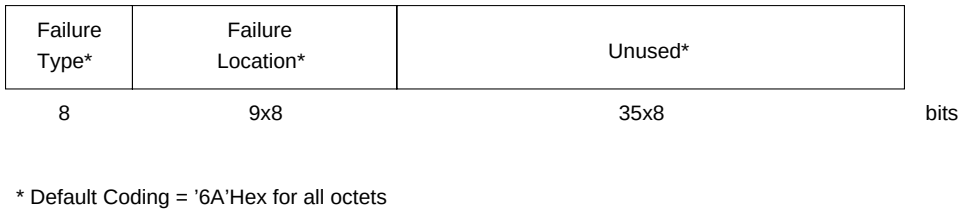

Abbildung 66. Funktionsspezifische Felder von AIS und RDI/FERF OAM-Zellen

Abbildung 67 zeigt, wie der Einsatz von AIS und RDI/FERF OAM-Zellen funktioniert. Im ersten Beispiel tritt nur in einer Richtung ein Fehler auf, im zweiten Beispiel in beiden Richtungen. Bemerkt im ersten Beispiel der benachbarte Knoten (Knoten 3) den Fehler so schickt er flußabwärts eine AIS-Zelle zum Verbindungsendpunkt (Knoten 4). Dieser schickt dann flußaufwärts eine RDI/FERF-Zelle zum anderen Endpunkt (Knoten 1). Im zweiten Beispiel sind beide Richtungen vom Fehler betroffen. Daher schicken auch beide benachbarten Knoten (Knoten 2 und 3) eine AIS-Zelle zum jeweiligen Verbindungsendpunkt (Knoten 1 und 4). Somit sind schon beide Endpunkte über den Fehler informiert. Nach Konvention schicken aber trotzdem beide eine RDI/FERF-Zelle zum entsprechenden anderen Endpunkt. Da in beiden Fällen beide Verbindungsendpunkte über den Fehler informiert sind, können sie durch ein anderes Routing die fehlerhafte Strecke umgehen.

3.2.2 Loopback Loopback-Zellen dienen dazu, Fehler aufzuspüren, die nicht durch das Prinzip der AIS und RDI/FERF-Zellen entdeckt werden können. Ein Beispiel hierfür ist eine Fehlkonfiguration der VPI- und/oder der VCI-Umsetzung, so daß die Zellen ihr Ziel nie erreichen. In Abbildung 68 sind die funktionsspezifischen Felder der Loopback OAM-Zellen zu sehen. Die einzelnen Felder haben folgende Bedeutung:

- *Loopback Indication* : Sie enthält "01", wenn die Zelle erzeugt wird, und wird vom Empfänger auf "00" gesetzt. Somit wird verhindert, daß die OAM-Zelle in eine Endlosschleife läuft.

- *Correlation Tag* : Es wird vom Erzeuger benutzt, um die zurückkommenden OAM-Zellen identifizieren zu können, da auf einem VPC/VCC mehrere Zellen unterwegs sein können.

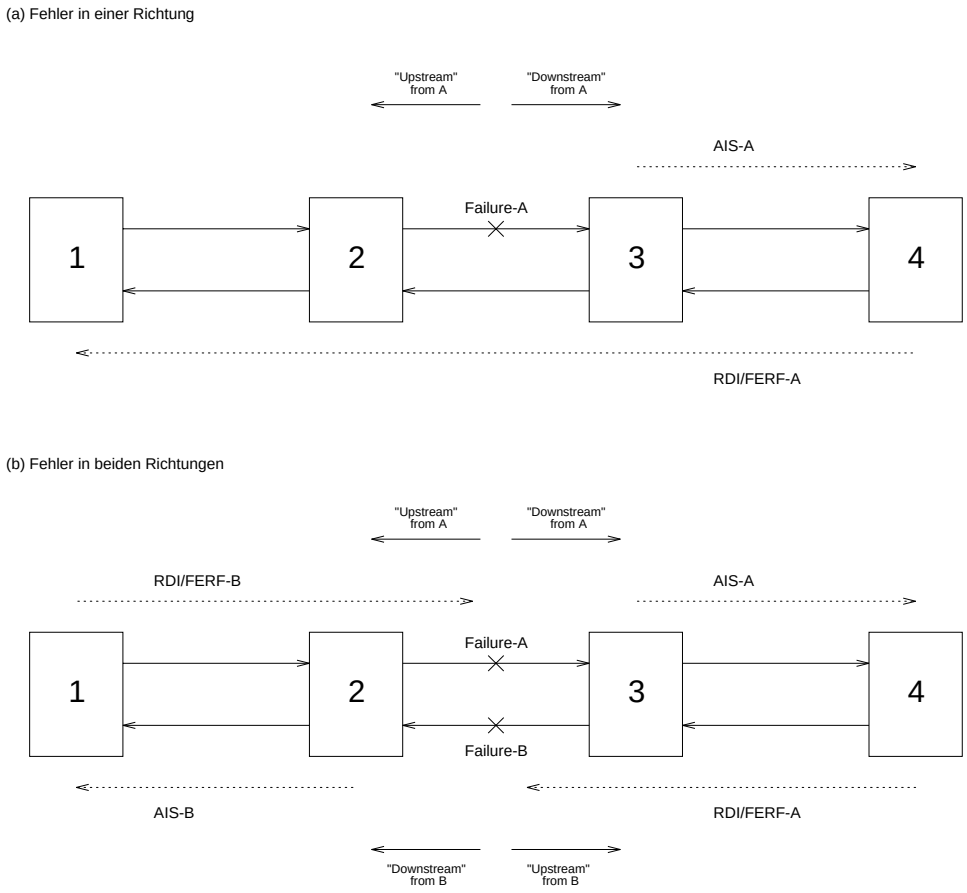

Abbildung 67. Funktionsweise von AIS und RDI/FERF OAM-Zellen

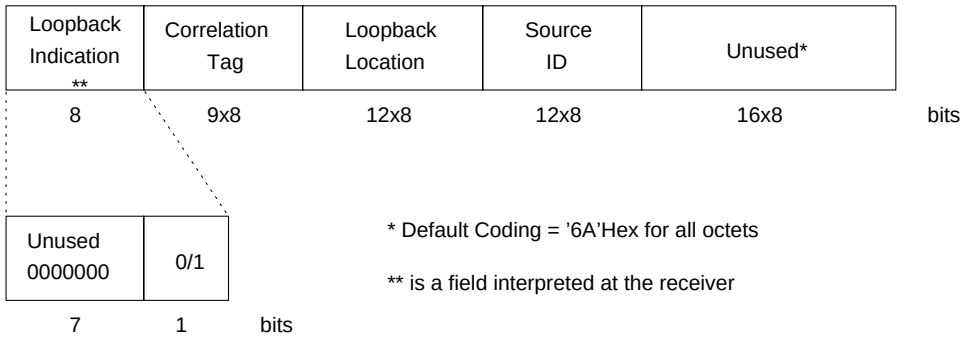

Abbildung 68. Funktionsspezifische Felder von Loopback OAM-Zellen

- *Loopback Location ID* : Sie zeigt bei einem Segment Loopback an, bei welchem Knoten der Loopback stattfinden soll. Der Default-Wert (alle Bits auf 1) legt fest, daß der Loopback beim Endpunkt stattfinden soll.
- *Source ID* : Sie wird benutzt, um die Loopback-Quelle feststellen zu können.

In Abbildung 69 sind Beispiele für die beiden grundlegenden Loopback Funktionen Segment-Loopback und Ende-zu-Ende-Loopback zu sehen. Im ersten Fall testet dabei Knoten 1 das Segment bis hin zu Knoten 3 und im zweiten Fall führt Knoten 1 ein Ende-zu-Ende-Loopback mit Endpunkt 2 durch.

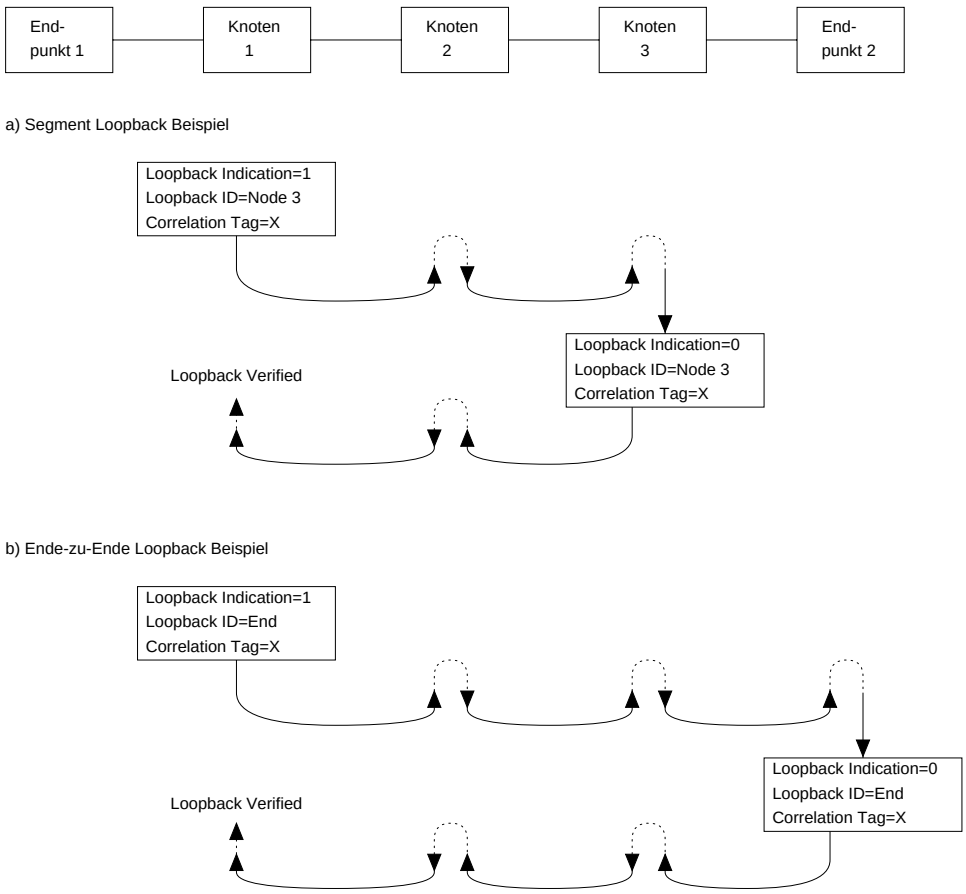

Abbildung 69. Beispiele für Loopback Funktionen

Abbildung 70 zeigt in einem weiteren Beispiel die Verwendung der Loopback-Funktion. In Teil a) wird ein Ende-zu-Ende-Loopback durchgeführt, um die Verbindung der beiden Endpunkte zu prüfen. Falls dieser fehlschlägt, kann z.B. das Netzwerk 2 den Fehler wie folgt finden. Teil b) zeigt die Überprüfung der Verbindung zwischen einem Knoten in Netzwerk 2 und dem Endpunkt 2 mit einem Ende-zu-Ende Loopback. Falls dieser fehlschlägt, liegt der Fehler zwischen Netzwerk 2 und Endpunkt 2. Teil c) zeigt das gleiche für Endpunkt 1. Falls dieser Loopback-Test fehlschlägt, liegt der Fehler entweder in der Verbindung zwischen Netzwerk 1 und Endpunkt 1, im Netzwerk 1, in der Verbindung zwischen Netzwerk 1 und 2 oder im Netzwerk 2 selbst. Teil d) zeigt die Überprüfung der Verbindung über Netzwerk 1 und Netzwerk 2 mit Hilfe eines Segment-Loopback-Tests. Falls dieser erfolgreich ist liegt der Fehler in der Verbindung zwischen Netzwerk 1 und Endpunkt 1. Teil e) zeigt die Überprüfung der Verbindung vom Eingang zum Ausgang in Netzwerk 2. Ist diese erfolgreich, so liegt der Fehler in Netzwerk 1.

3.2.3 Kontinuitätsüberprüfung Die Kontinuitätsüberprüfung dient dazu, mit Hilfe von periodisch in einem bestimmten Intervall gesendeter Zellen zwischen einer unbenutzten und einer fehlgeschlagegen Verbindung zu unterscheiden. Im ANSI Dokument [ANS94] wird dieses Konzept erweitert, indem festgelegt wird, daß in einem bestimmten Intervall (2 bis 20s) mindestens eine Nutz- oder Kontinuitätsüberprüfungs-Zelle empfangen werden muß. Andernfalls wird stromaufwärts eine VP-RDI/FERF-Zelle geschickt.

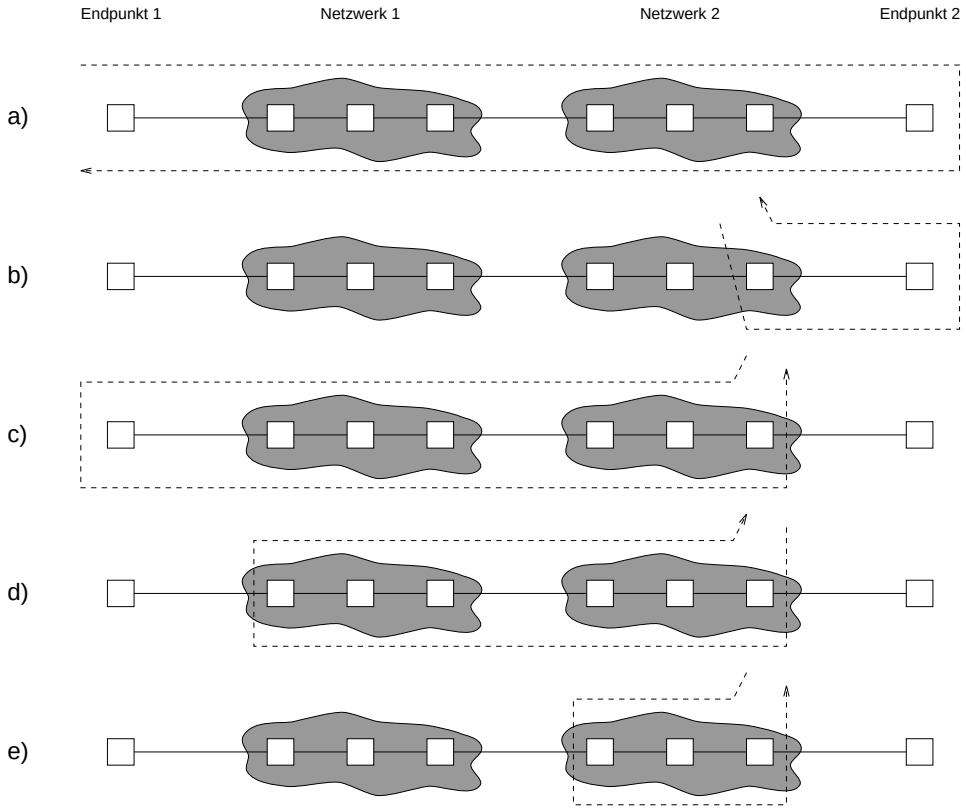

Abbildung 70. Anwendung der Loopback Funktion

Für die Kontinuitätsüberprüfungs-Zellen sind noch keine funktionsspezifischen Felder standardisiert, und sie sind lediglich für virtuelle Pfade (VP) gedacht.

Abbildung 71 gibt ein Beispiel für einen Fehler, der durch AIS nicht erkannt wird, wohl aber durch Kontinuitätsüberprüfung. Teil a) zeigt eine VPC über drei Cross-Connect-Knoten, die zur Zeit ausschließlich Kontinuitätsüberprüfungs-Zellen (CC) transportiert. In Teil b) findet eine fehlerhafte Vermittlung (Cross-Connect) in Knoten 2 statt, die den Strom der CC-Zellen abreißen läßt. In Teil c) stellt Knoten 3 den Kontinuitätsfehler fest und schickt eine VP-RDI/FERF-Zelle in entgegengesetzter Richtung.

3.3 Leistungsmanagement

Dieser Abschnitt beschäftigt sich mit der Messung und Einschätzung der Netzwerk-Leistung (Network Performance NP) und der Dienstqualität (Quality of Service QoS). Während QoS 'nur' die vom Benutzer am Endpunkt wahrgenommene Leistung des Dienstes darstellt, findet NP Verwendung im Netzmanagement.

3.3.1 NP/QOS Messung Abbildung 72 zeigt die funktionsspezifischen Felder der OAM-Zellen zum Aktivieren bzw. Deaktivieren der Leistungsmessung. Die einzelnen Felder haben folgende Bedeutung:

- *Message ID* :
  - '000001' Aktivierung (Anfrage)

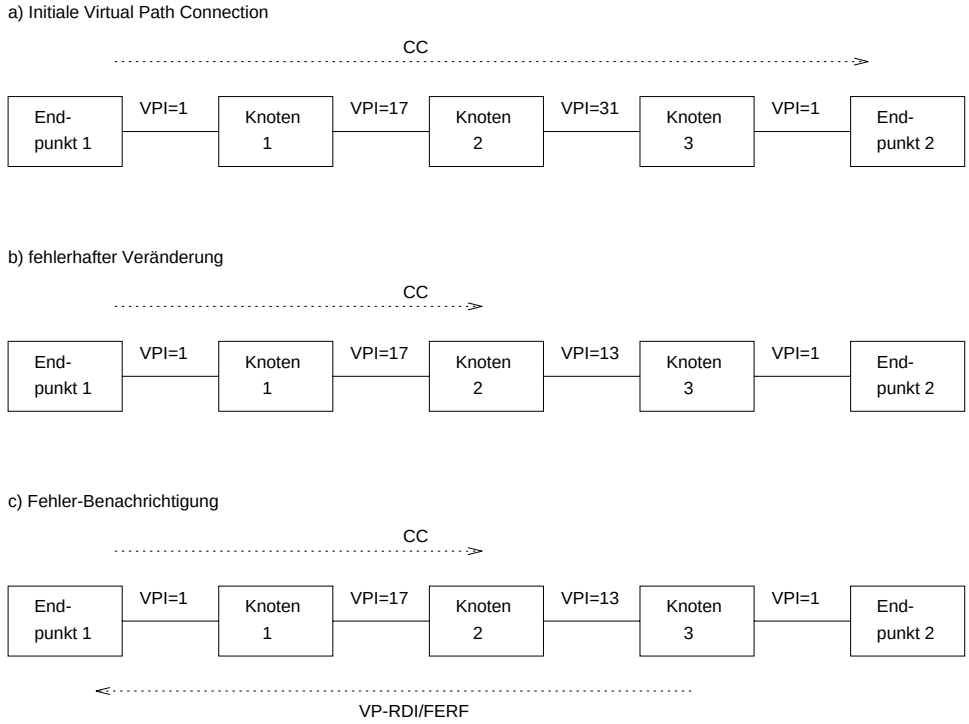

Abbildung 71. Verwendung von Kontinuitätsüberprüfungs-Zellen (CC)

- '000010' Aktivierung (Bestätigung)
- '000011' Aktivierung (Ablehnung der Anfrage)
- '000101' Deaktivierung (Anfrage)
- '000110' Deaktivierung (Bestätigung)
- '000111' Deaktivierung (Ablehnung der Anfrage)

– *Direction of Activation* : Sie ist definiert als A-B ('10') vom Aktivierer aus, B-A ('01') zum Aktivierer hin und ('11') für beide Richtungen.

– *Correlation Tag* : Es wird vom Erzeuger benutzt, um die zurückkommenden OAM-Zellen identifizieren zu können, da auf einem VPC/VCC mehrere Zellen unterwegs sein können.

– *PM Block Size A-B* : Sie bezeichnet die Größe des Blocks zur Leistungsmessung (Performance Measurement PM), der in A-B-Richtung unterstützt wird, kodiert durch eine Bitmaske für die Größen 1024, 512, 256 und 128.

– *PM Block Size B-A* : Sie bezeichnet in gleicher Art die Größe in B-A-Richtung.

Die Aktivierung/Deaktivierung läuft nach dem folgenden Schema ab. A schickt eine Aktivierungs-(Deaktivierungs-)Anfrage für A-B, B-A oder beide Richtungen an B. B schickt dann entweder eine Bestätigung oder eine Ablehnung der Anfrage zurück an A. Eine Ablehnung kann dadurch begründet sein, daß der Endpunkt nicht in der Lage ist, Leistungsmanagement-Funktionen zu unterstützen, oder daß die angeforderte Funktion

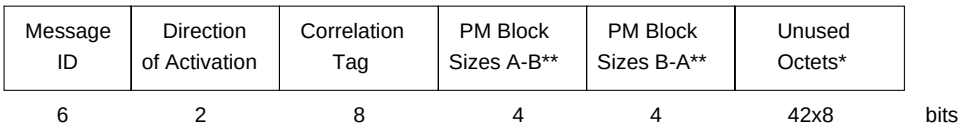

Abbildung 72. Funktionsspezifische Felder der Aktivierungs/Deaktivierungs-OAM-Zellen

im Moment nicht unterstützt wird. Falls die Anfrage beide Richtungen fordert, aber nur eine angeboten werden kann, so wird die Anfrage mit dem Hinweis bestätigt, daß die nichtausführbare Funktion ignoriert wird. Wenn die Leistungsmessung einmal aktiviert ist, läuft der im folgenden beschriebene Vorgang ab.

Abbildung 73 zeigt die dabei relevanten funktionsspezifischen Felder der OAM-Zellen zur Leistungmessung. Die einzelnen Felder haben folgende Bedeutung:

- *Monitoring Sequence Number (MSN)* : Dies ist die PM-Zellennummer modulo 256.
- *Total User Cell (TUC) Number* : Dies ist die absolute Zahl an Zellen mit Nutzdaten, die seit der letzten PM-Zelle gesendet wurden.
- *BIP-16* : Dies ist ein Block-Fehler-Code, der über alle Nutzzellen berechnet wird, die seit der letzten PM-Zelle gesendet wurden.
- *Time Stamp* : Er wird beim Backward Reporting benutzt, um die Verzögerung zu ermitteln. Dieser Zeitstempel sollte eine Genauigkeit von mindestens $1\mu s$ haben.
- *Block Error Result* : Dies wird beim Backward Reporting verwendet, um die Zahl der fehlerhaften BIP-16 Bits mitzuteilen.
- *Lost/Misinserted Cells* : Dies ist eine vorzeichenbehaftete Zahl, welche die Anzahl der empfangenen Zellen minus der TUC-Anzahl anzeigt.

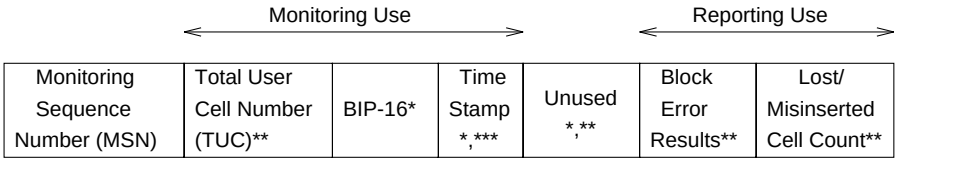

Abbildung 73. Funktionsspezifische Felder der OAM-Zellen zur Leistungsmessung

Abbildung 74 zeigt das Einfügen und Verarbeiten von OAM-Zellen zur Leistungsmessung. In beiden Richtungen werden hierfür PM-Zellen eingefügt. Dabei kann die Blockgröße der beiden Richtungen unterschiedlich groß sein. Auf der Senderseite werden die

PM-Zellen eingefügt, die Nutzzellen gezählt und der BIP-16 Wert berechnet. Auf der Empfängerseite werden die PM-Zellen dann wieder extrahiert, ebenfalls die Nutzzellen gezählt und der BIP-16 Wert berechnet, um sie mit den Werten der Senderseite zu vergleichen.

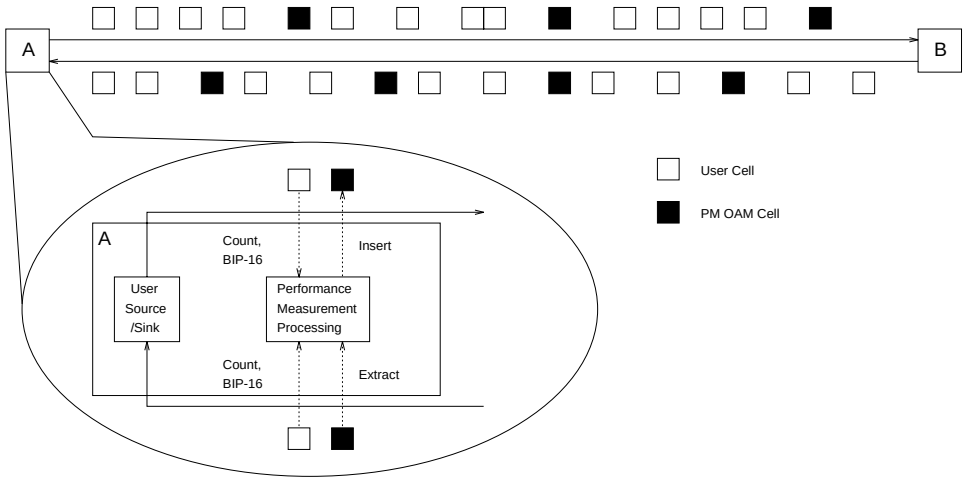

Abbildung 74. Funktionsweise der OAM-Leistungsmessung

3.3.2 NP/QOS Parameter Einschätzung In der ITU-T Empfehlung I.356 werden folgende Übertragungsergebnisse für Zellen definiert :

- *Erfolgreiche Zellenübertragung* : Die Zelle wird in richtiger Reihenfolge innerhalb einer bestimmten Zeit Tmax empfangen. Der binäre Inhalt der empfangenen Zelle weist keine Fehler auf und die Zelle besitzt einen gültigen Zellenkopf.
- *Fehlerhafte Zellenübertragung* : Die Zelle wird in richtiger Reihenfolge innerhalb einer bestimmten Zeit Tmax empfangen. Der binäre Inhalt der empfangenen Zelle weist Fehler auf oder die Zelle wird mit einem fehlerhaften Zellenkopf empfangen.
- *Zellenverlust* : Es wird innerhalb von Tmax keine Zelle empfangen.
- *Fehlplazierte Zellen* : Hierbei handelt es sich um Zellen, die in falscher Reihenfolge empfangen werden. Hierzu gehören sowohl in der Reihenfolge vertauschte Zellen als auch Phantom-Zellen.
- *Stark fehlerhafter Zellenblock* : Falls M oder mehr fehlerhafte Zellenübertragungen, Zellenverluste oder fehlplazierte Zellen in einem Zellenblock mit N fortlaufend übertragenen Zellen beobachtet werden, ist ein stark fehlerhafter Zellenblock gegeben.

Aus diesen Werten können die folgenden Leistungsmerkmale berechnet werden. Die Angaben in Klammern entsprechen dabei den zugehörigen QoS-Merkmalen.

Zellen-Fehler-Verhältnis	(Genauigkeit)
Verhältnis der stark fehlerhaften Zellenblöcke	(Genauigkeit)

Zellen-Verlust-Verhältnis	(Zuverlässigkeit)
Rate der fehlplazierten Zellen	(Genauigkeit)
Verzögerung der Zellenübertragung	(Geschwindigkeit)
Mittlere Verzögerung der Zellenübertragung	(Geschwindigkeit)
Variation der Verzögerung der Zellenübertragung	(Geschwindigkeit)

Im folgenden werden die aufgeführten Leistungsmerkmale genauer beschrieben und Anmerkungen zu ihrer Berechnung gegeben.

$$\text{Zellen-Fehler-Verhältnis} = \frac{\text{Fehlerhafte Zellen}}{\text{Erfolgreich übertragenen Zellen} + \text{Fehlerhafte Zellen}}$$

Erfolgreich übertragene Zellen aus stark fehlerhaften Zellenblöcken sollten dabei von der Berechnung des Zellen-Fehler-Verhältnisses ausgeschlossen werden.

Fehlerhafte Zellen können nur durch Zählen der bis zu M ($2 \leq M \leq 16$, mit einem Standartwert von 4) Paritätsfehler im BIP-16 Feld des Blocks ermittelt werden.

$$\text{Verhältnis der stark fehlerhaften Zellenblöcke} = \frac{\text{Stark fehlerhafte Zellenblöcke}}{\text{Insgesamt übertragene Zellenblöcke}}$$

Ein stark fehlerhafter Zellenblock ist ein Zellenblock mit mehr als M ($2 \leq M \leq 16$, mit einem Standart-Wert von 4) BIP-16 Fehlern oder mehr als K ($2 \leq K \leq M$, mit einem Standart-Wert von 2) verlorenen oder fehlplazierten Zellen.

$$\text{Zellen-Verlust-Verhältnis} = \frac{\text{Verlorene Zellen}}{\text{Insgesamt übertragene Zellen}}$$

Verlorene und übertragene Zellen aus stark fehlerhaften Zellenblöcken sollten von der Berechnung des Zellen-Verlust-Verhältnisses ausgeschlossen werden.

Die Anzahl der verlorenen Zellen kann aus der Differenz der letzten beiden Total User Cell Numbers (TUC) minus der Anzahl der im aktuellen Block empfangenen Zellen ermittelt werden.

$$\text{Rate der fehlplazierten Zellen} = \frac{\text{Fehlplazierte Zellen}}{\text{Zeit-Intervall}}$$

Stark fehlerhafte Zellenblöcke sollten von der Berechnung der Rate der fehlplazierten Zellen ausgenommen werden.

Die Anzahl der fehlplazierten Zellen kann aus der Anzahl der empfangenen Zellen minus der Differenz der beiden letzten Total User Cell Numbers (TUC) ermittelt werden.

Die Verzögerung der Zellenübertragung ist definiert als die Zeit von dem Moment, in dem die Zelle die Quelle verläßt, bis zu dem Moment, wo sie bei der Senke eintrifft. Die zur Zeit vorgeschlagenen Methoden zur Bestimmung der Verzögerung sind optional und benutzen entweder einen Zeitstempel oder ein wohldefiniertes Test-Signal. Für die Zeitstempel-Methode ist es notwendig, genau synchronisierte Uhren zu haben. Dafür sind aber noch keine Methoden standardisiert. Die mittlere Verzögerung ergibt sich aus

dem Durchschnitt einer bestimmten Anzahl von Verzögerungsmessungen. Die Varianz der Verzögerung läßt sich berechnen, indem eine Quelle alle T Sekunden eine PM-Zelle abschickt und die Senke die Abstände zwischen den Zellen mißt und hiervon jeweils T subtrahiert.

4 MIB für ATM-Loopback-Tests

Der aus [NT96] stammende Vorschlag für eine MIB für Loopback-Tests in ATM Netzen wird im folgenden zunächst beschrieben und anschließend in das OAM-Konzept eingeordnet. Diese MIB trägt den Namen ATMTEST-MIB.

4.1 Beschreibung

Aus Abbildung 75 geht hervor, daß der Wurzelknoten *atmTESTMIB* der ATMTEST-MIB momentan unter dem experimental-Knoten der grundlegenden MIB-II eingeordnet ist. Dies begründet sich damit, daß es sich bei dieser MIB nur um einen Vorschlag handelt. Falls die MIB zum Standard wird, soll sie unter dem Knoten *atmMIBObjects* eingegliedert werden.

Unterhalb von *atmTESTMIB* befinden sich die Knoten *atmTESTMIBObjects* und *atmTESTMIBConformance*, wobei der erste die Objekte enthält, die benötigt werden, um Tests durchführen zu können. Der zweite wird hier nicht näher betrachtet. Der Knoten *atmTESTMIBObjects* gliedert sich wiederum in die Zweige *atmLoopbackTestGroup* und *atmEndptGroup* auf. Im Zweig *atmEndptGroup* wird festgelegt, ob ein ATM-Knoten Endpunkt eines VPs bzw. VCs ist. Diese Information wird bei den Segment-Loopback-Tests benötigt, um festzustellen, in welchem Knoten der Loopback stattfinden soll. Im Zweig *atmLoopbackTestGroup* befinden sich die Blätter *atmLoopbackID*, *atmLoopbackLocID*, atmLoopbackSrcID sowie der Zweig *atmLoopbackTestTypes*. Das Objekt *atmLoopbackID* dient dazu, das ATM-Device zu identifizieren. Der Default-Wert ist alle Bits auf 1, was einen Segment-OAM-Loopback-Test bedeutet. Das Objekt *atmLoopbackID* kann von anderen Devices benutzt werden, um ein Loopback einer ATM-OAM-Zelle in diesem Device zu erreichen. Das Objekt *atmLoopbackLocID* gibt an, wo der Loopback stattfinden soll. Der Default-Wert ist alle Bits auf 1, was einen Segment-OAM-Loopback-Test bedeutet. Das Objekt *atmLoopbackSrcID* ist ein optionaler Parameter, der die Quelle des Loopback-Tests angibt. Die acht Testtypen, die als Blätter an *atmLoopbackTestTypes* hängen, werden nachfolgend beschrieben.

- *atmLoopbackVpE2e* : Dies ist ein Ende-zu-Ende-Test für VPs.
- *atmLoopbackVcE2e* : Dies ist ein Ende-zu-Ende-Test für VCs.
- *atmLoopbackVpSegment* : Mit diesem Test kann ein Segment eines VPs getestet werden.
- *atmLoopbackVcSegment* : Mit diesem Test kann ein Segment eines VCs getestet werden.

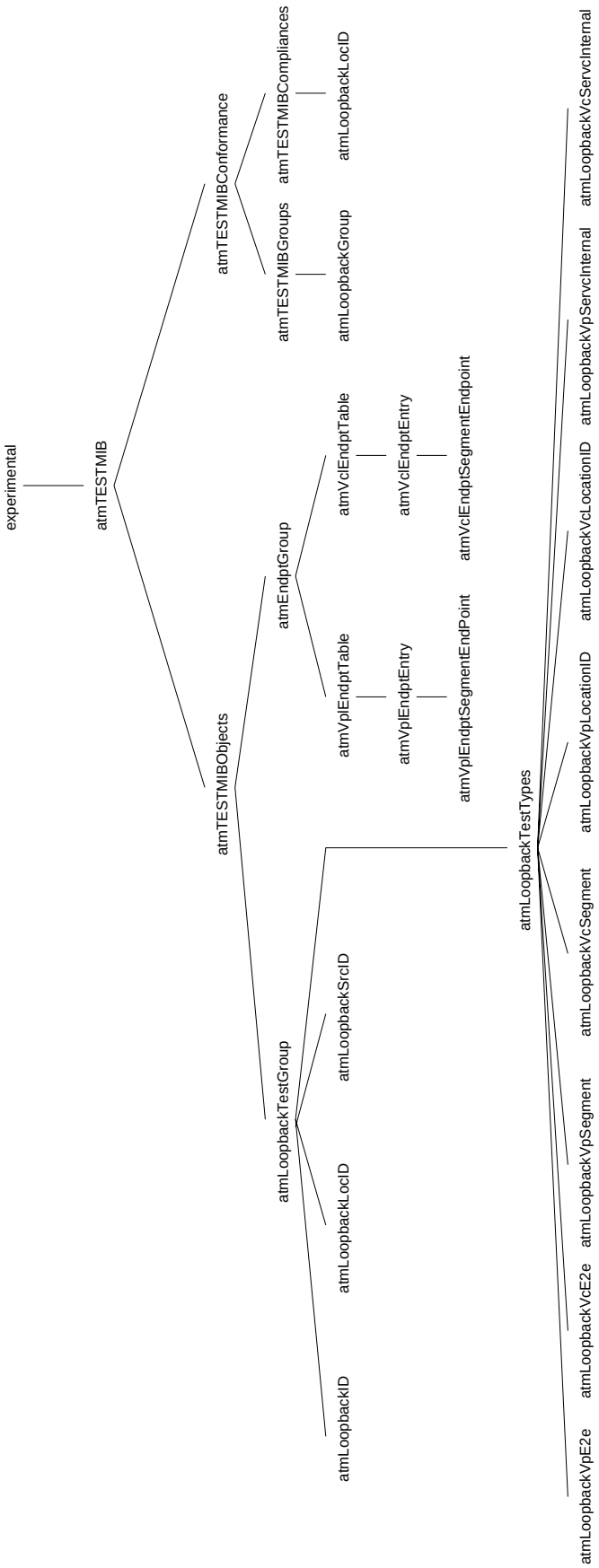

Abbildung 75. Die Test-MIB im Namensbaum für Managed Objects

– *atmLoopbackVpLocationID* : Dieser Test ermöglicht es, nur einen Teil eines VP Seg-

ments zu testen.

- *atmLoopbackVcLocationID* : Dieser Test ermöglicht es, nur einen Teil eines VC Segments zu testen.
- *atmLoopbackVpServcInternal* : Dieser Test veranlaßt den entsprechenden Agenten, den internen Teil eines bestimmten VPs zu testen.
- *atmLoopbackVcServcInternal* : Dieser Test veranlaßt den entsprechenden Agenten, den internen Teil eines bestimmten VCs zu testen.

Da nur ein Test zur Zeit auf einem Interface laufen kann, muß ein Manager zunächst den Besitz des Objekts *ifTestTable*, welches in [MK94] definiert wird, erhalten, um einen Test durchführen zu können. Als nächstes müssen die entsprechenden Parameter gesetzt werden. Für einen VP-Segment-Loopback-Test muß z.B. der Unterbezeichner *atmLoopbackVpSegment.p* mit dem VPI des zu testenden VPs belegt werden. Ausgeführt wird der Test dann, indem der Manager den Wert von *ifTestType*, welches ebenfalls in [MK94] definiert wird, mit dem Wert von *atmLoopbackVpSegment* belegt. Nachdem der Test gestartet wurde, erhält man das Ergebnis des Tests durch pollen des Objekts *ifTestResult*. Ist der Test noch nicht beendet, enthält *ifTestResult* den Wert *inProgress(3)*. Trifft die OAM Loopback-Zelle innerhalb von 5 Sekunden wieder beim Absender ein, so wird dies durch den Wert *success(2)* angezeigt. Andernfalls erhält *ifTestResult* den Wert *failed(7)*.

4.2 Einordnung in das OAM-Konzept

Über die ATMTEST-MIB können mit Hilfe von OAM-Zellen Loopback-Tests in ATM-Netzen durchgeführt werden. Die ATMTEST-MIB stellt somit eine Art Front-End für OAM-Loopback-Tests dar. Initiert eine Manager-Station über SNMP einen Loopback-Test bei einem Agenten, so verschickt dieser die entsprecheden OAM-Zellen und stellt je nach Ausgang des Tests das Ergebnis als MIB-Objekt zur Verfügung.

5 Zusammenfassung

Im Rahmen dieses Beitrages wurden das allgemeine OAM-Konzept zur Verwaltung von ATM-Netzen sowie die bisher zur Verfügung stehenden Funktionen zum Fehler- und Leistungsmanagement vorgestellt. Anschließend wurde anhand der ATMTEST-MIB gezeigt, wie man die OAM-Funktionen (in diesem Fall die Loopback-Tests) in der Praxis nutzen kann.

Da OAM bereits beim Entwurf des ATM-Standards voll integriert wurde, lassen sich auch nachträglich ohne große Probleme zusätzliche OAM-Funktionen hinzufügen bzw. die schon vorhandenen in ihrem Funktionsumfang erweitern. Dies ist ein großer Vorteil gegengüber dem ursprünglichen ISO/OSI-Referenzmodell, das keinerlei Mangementfunktionalität vorsieht. Wie in Abschnitt 3.1 zu sehen ist, sind auch längst noch nicht alle Kodierungen von OAM-Typen sowie deren Funktionstypen belegt. Somit bleiben noch viele Möglichkeiten zur Erweiterung von OAM offen.

Accounting in ATM-Netzen

Ulf Hansson

Kurzfassung

In diesem Beitrag werden zuerst die Anforderungen an ein Accountingmanagmentsystem vorgestellt, speziell bei großen Netzen und hohen Datenraten. Dann wird berichtet, welche Probleme heutige Managementsysteme in diesem Bereich noch aufweisen und was derzeit in verschiedenen Standardisierungsorganisationen diskutiert wird. Insbesondere wird das allgeimene Managementmodell des ATM-Forums vorgestellt sowie ein Vorschlag für eine Accounting-MIB, der von der Internet Engineering Task Force stammt.

1 Einleitung

Seit den ersten Schritten bei der Kommunikation zwischen Rechnern ist die Nachfrage nach besser Leistung und schnellerer Vermittlung stetig gestiegen und deshalb hat auch die Leistung und Vermittungsgeschwindigkeit zugenommen. Was damit begann, daß bequeme Betriebssystemsentwickler sich das Treppensteigen beim Datenaustausch sparen wollten, hat heute eine wichtige Rolle bekommen und viele Leute können sich ein Leben ohne ,,das Netz" nicht mehr vorstellen. Fast täglich entstehen neue Anwendungen für Netzwerke. Die nächste Generation der Netzwerke wird für Anwendungen wie Video auf Anforderung und Übertragung von medizinischen Daten in Echtzeit geplant und entworfen[Zit95].
Um diese zukünftigen Anforderungen an moderne Kommunikationsnetze und noch andere wie mehrere Teilnehmer und schnelleren Verkehr erfüllen zu können, finden neue Techniken wie z.B. ATM, FDDI und DQDB Verwendung[BHK95][Zit95][Tan96].
Die Nachfrage nach leistungsfähigeren Netzwerken hat auch die Forderungen im Bezug auf das Management erhöht. Obwohl in den letzten Jahren bereits viel über systemübergreifendes Netzwerkmanagment diskutiert wurde, sind noch keine endgültligen Lösungen in Sicht. Die meisten bereits existerenden Managementfunktionen sind problemspezifisch und deswegen nicht einfach erweiterbar [AC95]. Deswegen versuchen die verschiedenen Standardisierungsorganisationen, wie z.B. das ATM Forum, die ITU-T und die Internet Engineering Task Force, einen zukünftigen Netzwerkmanagmentstandard zu entwerfen. Dieses zukünftige Managementmodell sollte unter anderem verschiedene Möglichkeiten bei der Abrechnung auf dem Gebiet des Accounting und bessere Hilfsmittel zur Aufzeichnungen des Datenverkehrs bieten.

2 Grundlagen

2.1 Asynchroner Transfermodus - ATM (Asynchronous Transfer Mode)

Mit der Einführung des asynchronen Transfermodus ATM soll die immer stärker werdende Forderung nach Diensteintegration erfüllt werden. Um diese Diensteintegration und

andere Anforderungen zu erfüllen, verwendet ATM eine Technik, die auf Vermittlung von relativ kleinen Blöcken fester Länge, den sogenannten ATM-Zellen, basiert. Neben dieser Vereinfachung ermöglicht die moderne Technik mit ihrer Übertragungstechnik und den Lichtwellenleitern, die genannten Anforderungen zu erfüllen. Ein anderer wichtiger Begriff, der ebenfalls eine Vereinfachung darstellt, ist der Begriff der virtuellen Netzverbindungen (Virtual Channel)[Zit95]. Die prinzipiell zugrundeliegende Idee bei ATM ist aber nicht neu sondern stammt aus den 60er Jahren, damals unter den Namen Asynchronous Time Division Multipexing. Die Technik konnte sich aber zu der Zeit nicht durchsetzen, hauptsächlich wegen der leistungsschwachen Rechner und fehlender technischer Fortschritte im Bereich der integrierten Schaltkreise, und kam deshalb erst 1987 wieder in die Diskussion, als STM (Synchrone Transfermodus) die erwünchten Forderungen nicht erfüllen konnte. Die kurze, feste Zellenlänge macht ATM flexibel und schnell. Die kurze Zellenlänge sorgt dabei für die gewünchte Flexibilität beim Multiplexen und die feste Größe der Zellen vereinfacht die Verarbeitung der Zellen, so daß diese schneller durchgeführt werden kann. Der wichtige Bergriff der virtuellen Verbindungen bei ATM wurde bereits eingeführt. Wenn eine solche Verbindung aufgebaut ist, folgen alle ATM-Zellen der Verbindung dem gleichen Weg durch das Netz. Ein hierarchisches Verbindungskonzept unterschiedet dabei zwei unterschiedliche Typen von virtuellen Verbindungen: virtuelle Kanäle (Virtual Channel Connections, VCCs) und virtuelle Pfade (Virtual Path Connections, VPCs)[AT94].
Ein virtueller Kanal stellt eine unidirektionale Transportmöglichkeit zu Verfügung, obwohl es die Möglichkeit gibt, dem Fluß in einer Richtung eine Null-Kapazität zu geben.
Virtuelle Pfade hingegen repräsentieren eine Menge von virtuellen Kanälen mit den

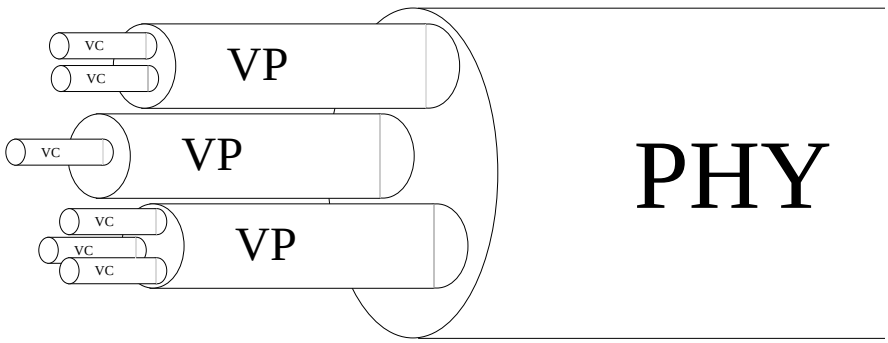

Abbildung 76. ATM-Verbindung. Die physische Verbindung ist in mehrere Pfade unterteilt, ein Pfad in mehrere Kanäle.

gleichen Endpunkten. Bei den virtuellen Verbindungen gibt es zwei verschiedene Typen, die permanenten virtuellen Verbindung (PVC) und die geschalteten virtuellen Verbindung (SVC) (Abbildung 76). Eine permanente Verbindung muß über das Management eingerichtet werden, eine geschaltete hingegen kann nach Bedarf mit Hilfe der Signalisierung aufgebaut werden. Beide Arten von Verbindungen können Punkt-zu-Punkt oder Punkt-zu-Mehrpunkt sein.

2.2 Management

In den frühen Tagen des Internets schickte man ,,pings“ hin und her und konnte von der Dauer der Antwort ableiten, wo Probleme bei der Datenübertragung waren. Mit zunemender Größe des Netzes war dieses Verfahren bald nicht mehr realistisch einsetzbar. Um für diesen Zweck jetzt etwas übersichtliches und praktisch einsetzbares zu erhalten, entwarf die Internet Engineering Task Force 1987 das Simple Gateway Monitoring Protocol (SGMP)[Sei94][BHK95].
Dieses Protokoll hatte als Ziel, die Information in den Gateways des Internets verwalten zu können. Die Teilziele dabei waren:

- geringe Kosten für die das Protokoll unterstützende Software;
- größtmögliche Ausnutzung der zu Verfügung stehenden Internet-Resourcen;
- möglichst geringe Beschränkung hinsichtlich der Managementverwendung;
- Realisierung einer leicht verständlichen und kompakten Menge von Managementfunktionen.

Die Managementinformation beim Internetmanagment liegt auf den ersten Blick genau wie beim ISO/OSI-Management in Objekten vor [BHK95], obwohl die Objekte des Internetmanagments [AT94], die Managed Objects, sich bezüglich das Aufbaus von denen des ISO-Managementrahmenwerks unterschieden. Die Objekte des Internetmanagments sind eigentlich nur Variablen ohne Zustände und nur mit Lese- und Schreiboperationen. Sie sind in einem Namensbaum (Abbildung 77) eingeordnet und können gelesen oder gesetzt werden. Die Struktur erlaubt auch Listen und Tabellen und wird gemäß dem Objekt-Type-Macro, das von der Structure of Management Information (SMI) beschreiben wird, definiert. Dabei werden die Objekte mittels ASN.1 spezifiert.
Jeder angeschlossenen Netzwerkeinheit ist eine Menge solcher Variabeln (Objekte) zugeteilt, die durch eine aktive Instanz auf dem Knoten, dem Agenten, für das Management zugänglich gemacht werden.
Der netzwerkweite Zugang zu den Information der Managed Objects in dem Namensbaum wird mit dem Simple Network Management Protocol (SNMP), einem einfachen Managementprotokoll, realisert. Das SNMP-Modell besteht dazu aus vier Teilen: die verwalteten Knoten, die Managementstationen, die Managementinformation und das Managementprotokoll [BHK95] [Tan96].

Ein SNMP-basiertes System bieten nicht die Möglichkeit, “das Ganze” zu überwachen; ein Netzwerkmanager will einen Ende-zu-Ende Überblick über z.B. Durchsatz und Jitter für alle virtuellen Verbindungen (bis zu zehntausend) haben. Er braucht die Möglichkeit, die unterliegende physikalische Struktur überwachen und kontrollieren zu können. Die neue ATM-Technik hat diese Schwäche noch deutlicher gezeigt. Manager von ATM-Netzen brauchen ein Hilfsmittel, ein Management- Framework, um Service-Provisioning, Fehler und Leistung, Kontrollverfahren und Resourcevergabe zu überwachen und zu steuern. Deswegen hat das ATM-Forum ein Modell entwickelt, das in fünf Schnittstellen

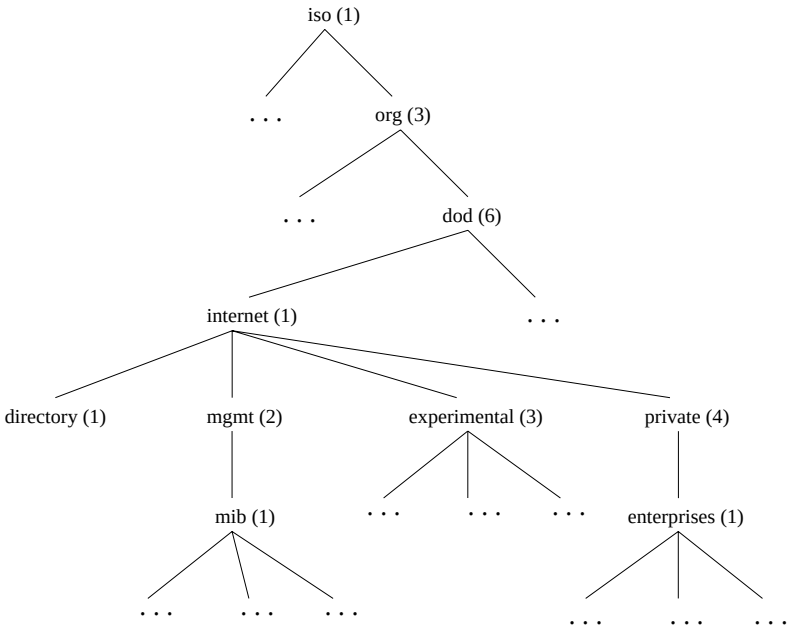

Abbildung 77. Der Namensbaum für Managed Objects

gegliedert ist [AC95]. Dieses Modell definiert Schnittstellen zwischen privaten und öffentlichen Netzwerken, zwischen LANs und WANs und es definiert auch die Funktionen, um Managementinformation automatisch im Netz zu verteilen.

Die Network Management Working Group (NMWG) des ATM-Forums hat ein übergreifendes Ende-zu-Ende-Management für private und öffentliche Netzwerke und ein Internetworking zwischen den Netzen entworfen. Fünf Schnittstellen für das Management sind dabei von der NMWG definiert worden, die *M1* bis *M5* genannt werden.
M1 und *M2* definieren die Schnittstellen zwischen einem Netzwerkmanagmentsystem des Kunden und einem ATM-Endsystem oder einem ATM-Switch.
M3, die Kundennetzwerkmanagmentschnittstelle (Customer Network Management Interface, CNM), beschreibt die Schnittstelle zwischen einem Kunden und dem Netzbetreiber. Diese Schittstelle ermöglicht es dem Kunden, Einsicht in das Netz des Betreibers (z.B. für eine Echtzeitkontrolle) zu erhalten.
M4 ist die Schnittstelle, die eine Zusammenarbeit zwischen dem Privatnetz und dem öffentlichen Netz mit Hilfe der Netzwerkmanagmentstufeschnittstelle (Network Management Level, NML) und dem Element Management Level (EML) ermöglicht. Diese beiden bieten zwei verschiedene Sichtweisen: zum einen vom Privatnetz ins öffenlichen Netz, um angebotene Dienste zu benutzen und Dienste bezüglich ihrer Qualität zu überwachen; zum anderen vom öffentlichen Netz ins Privatnetz, um als Diensterbringer herauszufinden, welche Dienste angeboten werden sollen.
M5 ist die Managementschnittstelle zwischen Netzwerkmanagmentsystemen des Diensterbringers. Sie ist die komplexeste Schnittstelle von den fünf genannten.

2.3 Accounting

Die Ende-zu-Ende-Architektur von ATM-Netzen erfordert, die Kostenzuteilung und Abrechnung zu verändern. Obwohl die Komplexität diese Aufgabe überwältigend ist, sollte schnell eine geeignete Lösung gefunden werden. Das erforderliche Berechnungssystem sollte den Diensterbringern (carriers) und den Netzanbietern (corporate networkers) die Möglichkeit geben, die Qualität für jede Verbindung zu bestimmen und darauf aufbauend die Benutzung durch die Anwendungen zu messen. Große Bereiche der Netzgemeinschaft sind der Meinung, daß anwendungsbasierte Berechnug die Lösung ist, um die Kosten für ATM billig zu halten und um ATM als einen Dienst für jederman anzubieten.
Anwendungsbasierte Berechnung bedeutet, daß der Benutzer nur für die verwendete Bandbreite bezahlt. Ebenso sind Dienste denkbar, bei denen die ausgelieferte Datenmenge abgerechnet wird, wie beispielsweise bei der Anforderung eines Videofilms (Video-on-Demand). Es ist auch beabsichtigt, daß Organizationen anwendungsbasierte Berechnung verwenden können, um zu bestimmen, wieviel die verschiedenen Abteilungen benutzen [AC95].

3 Vorschlag für eine Accounting-MIB

3.1 Motivation und Anforderungen

Nach Vorstellung der Arbeiten des ATM-Forum soll nun eine von der Internet Engineering Task Force vorgeschlagene MIB beschrieben werden.
Weil es sich dabei um einen experimentellen Internet-Draft handelt, ist noch nicht alles endgültig festgelegt. Die Information stammt jedoch aus dem zweiten Vorschlag für diese MIB [GP96], so daß bereits einige Diskussionsergebnisse eingearbeitet sind. Kurz vor der Fertigstellung dieses Beitrages ist sogar schon der dritte Vorschlag für diese MIB veröffentlicht worden [GMP96].
Das Ziel der MIB ist es, die erforderlichen Managed Objects zu bieten, die ein ATM-Accountingmanagment ermöglichen, und obwohl viele Netwerkmanager nur mit Information über den gesamten Verkehr umgehen, ist es manchmal auch nötig, Accountinginformation wie die Bandbreite einer Anwendung und andere Netzwerkmittel pro Verbindung messen zu können.
Die Möglichkeit zur Sammlung der Information in Dateien soll für alle Arten von Verkehr zu Verfügung stehen, sowohl für die permanenten virtuellen Verbindung (PVC) als auch für die geschalteten virtuellen Verbindung (SVC) und ebenso für die fast-permanenten virtuellen Verbindungen (SPVCCs).

Laut Internet Engineering Task Force soll die entworfene MIB die folgenden Grundlagen erfüllen:

- Ünterstützung für ATM PVC und SVC Daten.
- Das Speichern von PVC- und SVC-Daten muß nicht unterstützt werden, wenn es jedoch unterstützt wird, sollte es sich an die MIB-Spezifikation halten. Anbieter können dabei auch eigene Anpassungen machen.

- Das Speichern von PVC- und SVC-Daten sollte in getrennten Dateien geschehen. Allerdings sind dabei das gleiche Steuerverfahren und die gleichen MIB-Objekte zu verwenden.
- Das Ein- und Ausschalten des Speicherns in einer Log-Datei sollte durch die MIB ermöglicht werden. In dem Fall sollte auch die Trapinformation zu Veränderungen des Logging-Zustandes unterstützt werden.
- Die MIB sollte eine Untersuchung der Accounting-Information in der Logdatei sowie das Verändern der Struktur in dieser Datei durch Set-Komandos unterstützen.
- Der Zeitpunkt des Schreibens von Information in die Logdatei sollte flexibel eingestellt werden können. Entweder werden die Daten beim Abbau einer Verbindung in die Datei geschrieben, oder aber periodisch in einem vorbestimment Zeitintervall.
- Die Informationselemente für das Accounting sind flexibel zu beschreiben, damit zukünftige Erweiterungen einfacher realisierbar sind.
- Für Logdateien ist eine maximale Größe festzulegen, die eingehalten werden muß.
- Bei Erreichen der maximalen Größe einer Logdatei ist ein Trap zu erzeugen.
- Die Syntax (Low-Level-Kodierung) der Daten in einer Logdatei ist von der MIB festzulegen.
- Eine Operation sollte die Daten in die Logdatei schreiben.

Die Aufzeichnung der Accounting-Daten geschieht wie folgt:
Ein Netzwerkelement (Switch) sammelt die Daten zuerst in einem Zwischenspeicher. Wenn die Menge der Daten eine gewisse Obergrenze erreicht hat, schreibt er die Daten in eine Logdatei, die sich der Manager später holen kann.
Ein neues Konzept dieser MIB ist, SNMP nur für die Steuerung des Accountings zu verwenden. Die aufgezeichneten Accounting-Daten hingegen werden in einer Logdatei gespeichert, welche sich der Netzwerkmanager mit Hilfe von FTP oder einem ähnlichen Protokoll für den Dateiaustausch anschließend holen kann.

3.2 Die Objekte

Die Objekte der momentanen Version der Accounting-MIB sind zur Zeit im Namesbaum (Abbildung 77) unter "experimental" eingeordnet. Nach erfolgreicher Standardisierung wird die MIB dann wahrscheinlich unter mib(1) und accounting(12) eingeordnet. Die anderen Objekte unter mib(1) sind in Abbildung 78 aufgelistet. Das Objekt 9 wird nie benutzt und fehlt deswegen. Wenn eine Nummer nämlich einmal vergeben worden ist, kann sie nicht wiederverwendet werden.

Das gegenwärtige Modell der MIB sieht Log-Dateien für die Sammlung der Abrechnungsinformation vor. Eine solche Log-Datei hat eine maximale Größe, kann aber auch schon früher geschlossen werden. In folgenden Situationen wird eine Log-Datei geschlossen:

Nummer	*Bezeichnung*	*Erklärung*
1	system	Information über das zu verwaltende Gerät selbst
2	interfaces	Daten zum Netzwerk-Interface
3	at	Daten zur Adreßübersetzung zwischen IP-Adresse und physikalischer Adresse
4	ip	Daten über das Internet Protocol
5	icmp	Daten über das Internet Control Message Protocol
6	tcp	Daten über das Transmission Control Protocol
7	udp	Daten über das User Datagram Protocol
8	egp	Daten über das Extreior Gateway Protocol
10	transmission	Informationen über medienspezifische Objekte
11	snmp	Daten über das Simple Network Management Protocol

Abbildung 78. Teilbäume unter dem mib(1)-Knoten

- Eine Speicherung weiterer Daten würde die maximale Größe der Datei überschreiten.
- Der Netzwerkmanager startet das Loggen in eine neue Datei.
- Eine zeitbasierte Umschaltung sorgt für das Loggen in eine neue Datei.

Wenn eine Datei geschlossen worden ist, wird sofort eine neue Datei angelegt, in der ab diesem Zeitpunkt die neuen Abrechnungsdaten gespeichert werden. Der Name der neuen Datei unterscheidet sich von dem Namen der alten durch den um eins erhöhten ganzzahligen Suffix. Erst nach dem Schließen einer Datei ist es erlaubt, sie mit Hilfe von z.B. FTP zu holen.

Die Daten, die für jede Verbindung gesammelt werden, bestehen aus einer Menge von Objekten (Abbildung 79) und den Werten dieser Objekte. Für die Sammlung dieser Objekte und Werte gibt es zwei Anlässe: wenn eine SVC-oder PVC-Verbindung abgebaut wird und/oder periodisch für jede Verbindung.

Jede Zeile in der atmAcctngControlTable bietet die Möglichkeit, eine neue Datei zu spezifizieren, und das ermöglicht z.B. Daten über PVCs und SVCs in verschiedenen Dateien zu speichern.

Jede Datei enthält mindestens:

- Der Index der Kontrolldatentabelle.
- Welche Daten in der Datei gespeichert sind.
- Eine Beschreibung der nachfolgenden Objekte.
- Danach folgen null oder mehr Verbindungsdaten.

Im folgenden soll ein Beispiel das Zusammenspiel der Objekte in der MIB bei dem Schreiben von Accounting-Daten in ein Logfile verdeutlichen. Die MIB-Objekte haben dazu

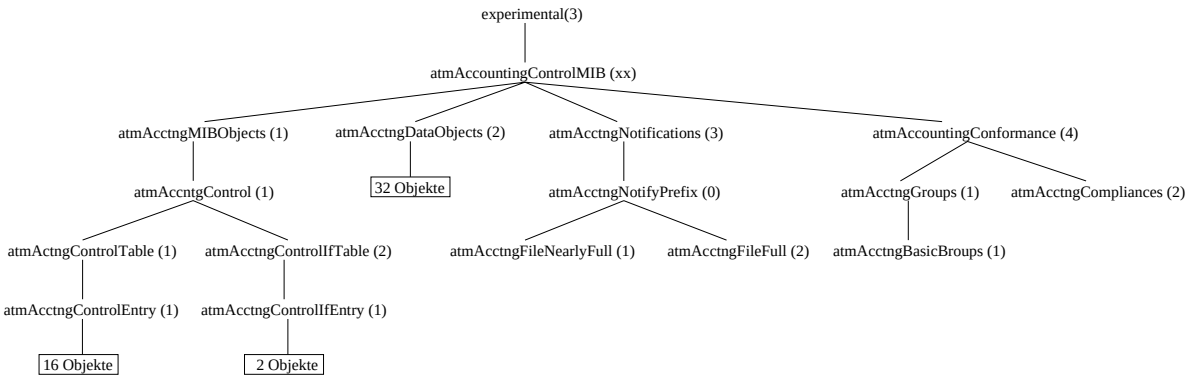

Abbildung 79. Die vorgeschlagene MIB

die folgenden Werte:

atmAcctngFilename: PVCData
atmAcctngFilenameCurrentSuffix: 1
atmAcctngCurrentSubtree, atmAcctngCurrentList: Zeigt auf PVC-zuhörige Objektelemente, die gespeichert werden sollen.
atmAcctngDescription: 'PVC-Daten für Switch XY'
atmAcctngCheckpointTimeInterval: 60 (Sekunden)
atmAcctngFileChangeTimeInterval: 86400 (Sekunden, d.h. 24 Stunden)
atmAcctngMaxFileSize: 1000000 (Bytes)

Die Objekte werden dann wie folgt verwendet:
Für das Speichern der Accounting-Daten wird eine Datei mit dem Namen PVCData.1 angelegt. Neue Daten werden jede Minute (d.h. alle 60 Sekunden) in die Datei PVCData.1 geschrieben. Alle 24 Stunden (86400 Sekunden) wird die aktuelle Datei geschlossen, das Suffix inkrementiert und eine neue Datei unter dem neuen Namen geöffnet. In diesem Beispiel würde die Datei PVCData.1 nach 24 Stunden geschlossen und eine Datei mit dem Namen PVCData.2 geöffnet. Wenn dann beispielsweise die Datei PVCData.2 innerhalb der nächsten 24 Stunden die maximale Größe von 1000000 Bytes überschreitet, wird schon vor Ablauf der festgelegten Frist für den Dateiwechsel (24 Stunden) die Datei PVCData.2 geschlossen und eine neue Datei mit dem Namen PVCData.3 geöffnet.

Wie Abbildung 79 zeigt, besteht die MIB aus vier "Hauptteilbäumen", die sich in weitere Bäume aufgliedern. Die wichtigste Teilbäume sind die atmAcctngMIBObjects (1) und die atmAcctngDataObjects (2). Diese enthalten alle wesentlichen Objekte, die beim Accounting benötigt werden.

atmAcctngTable
Die Tabelle atmAcctngTable ist ein Hauptbestandsteil der MIB und enthält folgende atmAcctngControlEnty-Einträge:

atmAcctngControlIndex (1) - Durch den Kontrollindex wird die Zeile eindeutig iden-

tifiziert.

atmAcctngCurrentSubtree (2) - Das Tupel (atmAcctngCurrentSubtree, atmAcctngCurrentList) beschreibt die zu überwachenden Objekte. Dabei legt atmAcctngCurrentSubtree den Teilbaum aus der MIB fest.

atmAcctngCurrentList (3) - Das Tupel (atmAcctngCurrentSubtree, atmAcctngCurrentList) beschreibt die zu überwachenden Objekte. Dabei legt atmAcctngCurrentList die Objekte aus dem Teilbaum fest.

atmAcctngFilename (4) - Der Hauptteil des Names der Datei, in der die Accountingdaten gerade gespeichert werden. Dieser Name wird noch um einen Suffix ergänzt, der in atmAcctngFilenameCurrentSuffix definiert ist.

atmAcctngFilenameCurrentSuffix (5) - Der Suffix des Names der Datei, in der die Accountingdaten gerade gespeichert werden. Dieser Suffix ist ganzzahlig und wird inkrementiert, wenn die Datei ihre maximale Größe erreichen würde oder wenn eine zeitbasierte Umschaltung zu einer neuen Datei erfolgt.

atmAcctngForceCheckpoint (6) - Dieses Objekt ermöglicht es, die aktuelle Datenmenge in eine Datei zu schreiben und dabei auch das aktuelle Suffix um eins zu erhöhen.

atmAcctngNextSubtree (7) - Mit dem Tupel (atmAcctngNextSubtree, atmAcctngNextList) können zusätzliche Objekte für die Überwachung definiert werden. Wenn keine zusätzliche Objekte überwacht werden sollen, sind atmAcctngNextSubtree und atmAcctngCurrentSubtree gleich.

atmAcctngNextList (8) - Mit dem Tupel (atmAcctngNextSubtree, atmAcctngNextList) können zusätzliche Objekte für die Überwachung definiert werden. Wenn keine zusätzliche Objekte überwacht werden sollen, sind atmAcctngNextList und atmAcctngCurrentList gleich.

atmAcctngDescription (9) - Eine Textbeschreibung zur Benennung und Erläuterung der aktuellen Zeile.

atmAcctngMaxFileSize (10) - Die maximale Größe der Datei, in der die Daten gespeichert werden. Wenn die Größe der Datei diesen Wert erreicht, werden die folgenden Daten verworfen oder die sie werden in einer neuen Datei mit einem erhöhten Suffixwert gespeichert. Die Größe ist veränderbar, aber nur in größere Werte.

atmAcctngFileEncoding (11) - Dieses Objekt legt das Format fest, in dem die Daten in der Datei gespeichert werden. Voreingestellt ist dabei das Format BER (Basic Encoding Rules).

atmAcctngCheckpointTimeInterval (12) - Dieses Objekt definiert periodische Kontrollpunkte, an denen dann neue Daten gespeichert werden. Mit Hilfe des Objekts wird dazu das Intervall kontrolliert und neu gesetzt. Wenn Kontrollpunkte nicht benutzt oder unterstützt werden, enthält das Objekt den Wert (-1).

atmAcctngFileChangeTimeInterval (13) - Für Systeme, die das periodische Speichern von Daten unterstützen, ermöglicht dieses Objekt das Lesen und Setzen des Intervalls (in Sekunden) zwischen dem Neuanlegen zweier aufeinanderfolgender Dateien. Wenn das periodische Speichern nicht benutzt oder unterstützt wird, enthält das Objekt den Wert (-1).

atmAcctngTrapThreshold (14) - Dieser Grenzwert legt die Größe der Datei fest, bei deren Erreichen ein Trap („nearly-full") zur Benachrichtigung an den Netzwerkmanager geschickt wird.

atmAcctngTrapEnable (15) - Mit Hilfe dieses Objekts wird festgelegt, ob die Traps atmAcctngFileNearlyFull und atmAcctngFilefull zur Zeit aktiviert sind.

atmAcctngRowStatus (16) - Dieses Objekt dient dazu, eine neue Zeile in der Tabelle anzulegen, eine bestehende Zeile zu ändern oder eine Zeile zu löschen.

atmAcctngControlIfTable
Die Tabelle atmAcctngControlIfTable verwaltet die Objekte zu den Interfaces (Schnittstellen des Geräts), an denen die Daten gesammelt werden sollen. Sie enthält folgende Einträge:

atmAcctngControlIfEnable (1) - Dieses Objekt zeigt an, für welche Arten von Verbindungen an dieser Schnittstelle (wenn überhaupt) Aufzeichnungen gemacht werden sollen.

atmAcctngControlIfCollectMode (2) - Dieses Objekt zeigt an, wann die Accountingdaten für diese Schnittstelle in die aktuelle Datei geschrieben werden sollen.

atmAcctngDataObjekts
Das Objekt atmAcctngDataObjekts stellt einen Teilbaum dar, der die 32 Objekte enthält, welche momentan für eine Aufzeichnung beim Accounting definiert sind und somit dem Benutzer zur Verfügung stehen. Sie werden im folgenden vorgestellt:

atmAcctngIfIndex (1) - Dies ist der Index für das Interface (Schnittstelle des Geräts), der auch in der ifTable (MIB-II) enthalten ist.

atmAcctngCurrentTimeStamp (2) - Dieser Zeitstempel definiert die aktuelle Zeit für die Daten.

atmAcctngSysName (3) - Dieses Objekt legt einen Namen für den Knoten fest. In der Regel entspricht dieser Name dem echten Knotennamen in der jeweiligen Umgebung.

atmAcctngConnectionType (4) - Hiermit wird die Art der Verbindung beschrieben, die PVC, PVP, SVC oder SVP sein kann.

atmAcctngCastType (5) - Dieses Objekt beschreibt den Typ einer Verbindung, der Punkt-zu-Punkt oder Punkt-zu-Mehrpunkt sein kann.

atmAcctngIfName (6) - Hierbei handelt es sich um den Namen der aktuellen Schnittstelle, für welche die Daten gesammelt werden. Wenn der lokale SNMP-Agent das Objekt ifName unterstützt, wird dieser Name benutzt.

atmAcctngVpi (7) - Dieses Objekt enthält den VPI (Virtual Path Identifier) der Verbindung.

atmAcctngVci (8) - Dieses Objekt enthält den VCI (Virtual Channel Identifier) der Verbindung.

atmAcctngCallingParty (9) - Eine Kennzeichnung des Initiators der Verbindung (des Anrufers) ist hier enthalten. Wenn der Gesprächspartner unbekannt ist, steht nur ein EOF-Zeichen in dem Objekt.

atmAcctngCalledParty (10) - Dieses Objekt identifiziert den zweiten Teilnehmer der Verbindung (den Angerufenen). Wenn der Gesprächspartner unbekannt ist, steht nur ein EOF-Zeichen in dem Objekt.

atmAcctngCallReference (11) - Hier ist eine Referenznummer für die Verbindung enthalten.

atmAcctngStartTime (12) - Der Zeitpunkt, ab dem die Verbindung erfolgreich aufgebaut war, ist in diesem Objekt festgelegt.

atmAcctngCollectionTime (13) - Hier wird der Zeitpunkt des Verbindungsabbaus festgehalten.

atmAcctngCollectMode (14) - Mit Hilfe dieses Objekts kann der Grund angegeben werden, warum Daten zu dieser Verbindung aufgezeichnet werden.

atmAcctngReleaseCause (15) - Dieses Objekt beschreibt den Grund für den Abbau der Verbindung.

atmAcctngServiceCategory (16) - Hier wird der Diensttyp der Verbindung festgelegt.

atmAcctngTransmittedCells (17) - Die Anzahl auf dieser Verbindung übertragener Zellen wird von diesem Objekt festgehalten.

atmAcctngTransmittedClp0Cells (18) - Die Anzahl auf dieser Verbindung übertragener Zellen mit CLP = 0 ist hier enthalten.

atmAcctngReceivedCells (19) - Die Anzahl auf dieser Verbindung empfangener Zellen wird durch dieses Objekt angegeben.

atmAcctngReceivedClp0Cells (20) - Die Anzahl auf dieser Verbindung empfangener Zellen mit CLP = 0 (Cell Loss Priority) ist hier enthalten.

atmAcctngTransmittTrafficDescriptorType (21) - Hier wird die Art des ausgehenden Verkehrs der Verbindung beschrieben. Der Wert dieses Objekts legt die Bedeutung der folgenden 5 Objekte fest.

atmAcctngTransmittTrafficDescriptorParam1 (22) - Dieses Objekt enthält den ersten Parameter zur Beschreibung des ausgehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngTransmitDescriptorType ab.

atmAcctngTransmittTrafficDescriptorParam2 (23) - Dieses Objekt enthält den zweiten Parameter zur Beschreibung des ausgehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngTransmitDescriptorType ab.

atmAcctngTransmittTrafficDescriptorParam3 (24) - Dieses Objekt enthält den dritten Parameter zur Beschreibung des ausgehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngTransmitDescriptorType ab.

atmAcctngTransmittTrafficDescriptorParam4 (25) - Dieses Objekt enthält den vierten Parameter zur Beschreibung des ausgehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngTransmitDescriptorType ab.

atmAcctngTransmittTrafficDescriptorParam5 (26) - Dieses Objekt enthält den fünften Parameter zur Beschreibung des ausgehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngTransmitDescriptorType ab.

atmAcctngReceiveTrafficDescriptorType (27) - Hier wird die Art des eingehenden Verkehrs der Verbindung beschrieben. Der Wert dieses Objekts legt die Bedeutung der folgenden 5 Objekte fest.

atmAcctngReceiveTrafficDescriptorParam1 (28) - Dieses Objekt beinhaltet den ersten Parameter zur Beschreibung des eingehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngReceiveTrafficDescriptorType ab.

atmAcctngReceiveTrafficDescriptorParam2 (29) - Dieses Objekt beinhaltet den zweiten Parameter zur Beschreibung des eingehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngReceiveTrafficDescriptorType ab.

atmAcctngReceiveTrafficDescriptorParam3 (30) - Dieses Objekt beinhaltet den dritten Parameter zur Beschreibung des eingehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngReceiveTrafficDescriptorType ab.

atmAcctngReceiveTrafficDescriptorParam4 (31) - Dieses Objekt beinhaltet den vierten Parameter zur Beschreibung des eingehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngReceiveTrafficDescriptorType ab.

atmAcctngReceiveTrafficDescriptorParam5 (32) - Dieses Objekt beinhaltet den fünften Parameter zur Beschreibung des eingehenden Verkehrs. Die genaue Bedeutung dieses Objekts hängt vom Wert des Objekts atmAcctngReceiveTrafficDescriptorType ab.

atmAcctngNotifyPrefix
Schließlich wird hier noch der Teilbaum atmAcctngNotifyPrefix vorgestellt, der relativ wenig Information beinhaltet.

atmAcctngFileNearlyFull (1) - Eine Anzeige, daß die Größe der aktuellen Logdatei für die Accountingdaten fast die maximale Größe erreicht hat. Diese Warngrenze kann in der MIB eingestellt werden.

atmAcctngFileFull (2) - Eine Anzeige, daß die maximale Dateigröße überschritten worden ist und somit entweder in eine neue Datei geschrieben wird oder die Daten verworfen werden. Diese Anzeige kann als Zeichen dienen, die alte Datei auf das Accountingsystem zu überführen (z.B. mit ftp).

4 Zusammenfassung

Schon seit mehren Jahren war klar, daß die zur Verfügung stehenden Möglichkeiten für das Management nicht für die immer größer werdenden Netze ausreichend sind. Dies beweisen vor allem die neuen Hochleistungsnetze wie ATM. Die momentan verfügbaren Werkzeuge sind zumeist für einzelne Zwecke ausgelegt und können daher in der Regel nicht zusammenarbeiten.
Nach Meinung des ATM-Forums ist deshalb ein ganz anderer, neuer Ansatz erforderlich. Zu diesem Zweck hat das ATM-Forum eigene Schnittstellen für das Management von ATM-Netzen definiert, die zukünftige Managementprobleme lösen sollen.
Inzwischen hat auch die Internet Engineering Task Force einen Vorschlag für eine zukünftige Lösung entworfen. Bezüglich des Accountings in ATM-Netzen ist mittlerweile der dritte Draft als Standardisierungsvorschlag veröffentlicht worden. Diese Accounting-MIB ermöglicht es, die gewünschten Daten von permanenten virtuellen Verbindungen (PVC), von geschalteten virtuellen Verbindung (SVC) sowie von fast-permanenten virtuellen Verbindungen (SPVCCs) in einer Logdatei abzulegen, die dann von Netzwerkmanager per Filetransfer abgeholt werden kann.

Telecommunications Information Networking Architekture

Björn Müller

Kurzfassung

Technische Entwicklungen, wie das Home Banking, schaffen neue Computeranwendungen und Dienste, die eine engere Beziehung zwischen der Telekommunikation und der Informatik erfordern. Der Artikel beschreibt die Architektur des Telekommunikations-Informations-Netzwerks, das zum Ziel hat, diesen neuen Anforderungen gerecht zu werden. Da dieses auf bestehenden Standards aufbaut, werden die Grundlagen des Intelligenten Netzwerks wie auch des Telekommunikations Management Netzwerks angeführt. Als ein Ergebnis der TINA Initiative wird das Netzwerk Ressourcen Informations Modell vorgestellt, das der Dienstsoftware die notwendige Abstraktion zur Verfügung stellt, um die Ressourcen des Netzwerks zu benutzen und zu verwalten.

1 Einleitung

Die bislang getrennten Gebiete der Telekommunikation und der Informatik werden durch neue Techniken immer stärker miteinander verbunden. Kommunikationsnetzwerke nutzen in zunehmendem Maße Computer, und Computer kommunizieren mehr und mehr mit Hilfe von Telekommunikationsnetzwerken.

Neue technische Entwicklungen wie Videokonferenzen, Home Banking und Computerarbeitsplätze schaffen neue Computeranwendungen und Dienste, die eine engere Beziehung zwischen der Telekommunikation und der Informatik erfordern.

Diese Anforderungen fördern den gegenseitigen Austausch von Ideen zwischen der Computer- und der Telekommunikationsindustrie. Desweitern haben Entwicklungen im Bereich des verteilten Rechnens und der Breitbandkommunikationssyteme die Machbarkeit von Telekommunikationsdiensten als verteilte Anwendungen erhöht.

Der Wunsch nach anspruchsvollen Diensten, wie z.B. universelle persönliche Kommunikation, virtuelle Netzwerkdienste und Multimediadienste nimmt zu. Diese Dienste erfordern einen flexibleren Zugang, flexibleres Management und eine flexiblere Abrechnung, als es heutige Dienste anbieten können. Sie benötigen eine Netzwerk-Infrastruktur, in welcher Dienste leicht, schnell und problemlos eingeführt werden können.

Der wichtigste Bestandteil dieser Infrastruktur ist die Software. Die Wiederverwendbarkeit, die Portabilität und die Mehrfachnutzung der Softwarekomponenten sind die wichtigsten zu lösenden Aufgaben.

Die hier vorgestellte Telekommunikations-Informations-Netzwerk Architektur versucht, diese Aufgaben zu lösen.

2 Management von Telekommunikationsnetzen

2.1 Anforderungen an moderne Telekommunikationsnetzwerke

Die Anforderungen an Telekommunikationsnetzwerke steigen mit dem technischen Fortschritt und der Anzahl und Komplexität ihrer Dienste. Ein modernes Informationsnetzwerk soll dem Benutzer die Zugriffsmöglichkeiten auf Informationen sowie das Bearbeiten von Information zu jeder Zeit, an jedem Ort, in jedem Umfang und in jeder Form ermöglichen.

Eine Informations-Netzwerk-Architektur soll wiederverwendbare Netzwerkfunktionen zusammenfassen und eine netzwerkweite Verwendbarkeit unterstützen.

2.2 Vom „Intelligenten Netzwerk“ zum „Informations-Netzwerk“

Das Ziel der Entwicklung eines Intelligenten Netzwerks (IN) ist die Konstruktion einer Menge von Schnittstellen und Protokollen, die eindeutig die Vermittlungs- von den Dienstaspekten trennen [JBI93]. Das Intelligente Netzwerk muß dem unter Umständen verteilten Charakter der Anwendungssoftware sowie der Mehrfachverwendung durch verteiltes Netzwerkmanagement, Dienstmanagement und CPE-Software gerecht werden. Aus diesen Gründen ist ein objektorientierter Ansatz für die IN-Architektur sinnvoll, da die Objektorientierung diese Prinzipien unterstützt.

Objekte bilden eine natürliche Grenze für die Verteilung und sind fundamental für die Entwicklung fortschrittlicher verteilter Prozeßsoftware. Die Zahl der neuen Dienste, wie Multimediadienste, nimmt stetig zu. Deshalb muß ein Verbindungsmodell flexible mehrfache Kanalkontrolle unterstützen. Ein Kommunikationsmodell muß für diese neuen Dienste konstruiert sein und es wird erwartet, daß solch ein Modell objektorientiert sein wird.

Um schnell komplexe Dienste einführen zu können, ist eine einheitliche verteilte Prozeßumgebung nützlich, die es Netzwerkknoten erlaubt, Dienste anzubieten, und es anderen Diensten erlaubt, auf diese Dienste zuzugreifen.

Vornehmlich aus diesen Gründen wird sich die heutige IN-Architektur zu einer Informations-Netzwerk-Architektur entwickeln. Dabei liegt die Konzentration auf einer fortschrittlichen Entwicklung vom physikalischen Netzwerk hin zu einem Software System mit den Prinzipien des verteilten Arbeitens der Software Architektur.

Ein Informations-Netzwerk Modell erfordert ein objektorientiertes Modell zur Entwicklung von verteilten Anwendungen, welches Prinzipien der Trennung unterstützt. Diese Prinzipien sind:

- Trennung zwischen Verbindungs- und Übertragungsprodukttechnologiedetails und Funktionen, die mit dem Dienst zu tun haben
- Trennung zwischen gemeinsamem Dateimanagement und der geschäftsspezifischen Nutzung dieser Daten
- Trennung zwischen Details der Benutzerschnittstellen und anderen Kerngeschäftsfunktionen

In den nächsten Abschnitten werden die Komponenten und Konzepte eines objektorientierten Modells beschrieben.

2.2.1 Objekte und Dienste Ein Objekt ist ein Teil des Telekommunikationssystems. Es umfaßt Daten und deren Verarbeitung. Dabei werden zwei Arten von Objekten unterschieden, die „rechnenden" Objekte und die „nichtrechnenden" Objekte.

Zu einem rechnenden Objekt gehört eine Menge von Operationen, die auf diesem Objekt definiert ist. Ein solches Objekt interagiert mit anderen rechnenden Objekten, indem es Operationen auf diesen aufruft. Ein Zugriff oder eine Veränderung der Datenkomponente ist nur durch eine der Operationen möglich, die auf diesem Objekt definiert sind. Beispiele für rechnende Objekte sind:

- Objekte, die den kürzesten Weg zwischen Quelle und Bestimmungspunkt einer Verbindung berechnen
- Objekte, die Managementoperationen auf einer physikalischen Schicht unterstützen

Nichtrechnende Objekte interagieren miteinander in allgemeinerer Art und Weise durch den Austausch von Information, die als Sequenz von Bits codiert ist. Beispiele für nichtrechnende Objekte sind:

- Bidirektionale Gesprächskommunikation
- Audio Video Kommunikation

Unter Diensten versteht man eine Menge von Fähigkeiten, die von einem Objekt angeboten oder unterstützt werden, das von anderen Objekten benutzt werden kann. Dienste können auch Endbenutzern angeboten werden (z.B. Konferenzschaltungen). Die Definition von Diensten, die durch Objekte zur Verfügung gestellt werden, besteht aus der Festlegung der Schnittstelle und der Dienstsemantik. Ein Objekt kann mehrere Dienste zur Verfügung stellen, welche jeweils eine andere Semantik haben. Zwei Objekte, die über einen Dienst interagieren, haben eine Dienstgeber/Dienstnehmer-Beziehung.

Jeder Dienst ist eine Instanz eines Diensttyps, wobei Dienste, die Instanzen desselben Diensttyps sind, identische Dienstschnittstellen und -semantik haben. Instanzen können sich in der Leistung, der Qualität der Dienstparameter, den Gebühren usw. unterscheiden. Diese Aspekte werden Dienstattribute genannt.

2.2.2 Building Blocks und Dienstverträge Ein Building Block ist ein Softwareprodukt, das mehr als ein rechnendes Objekt enthält. Ein solches Objekt innerhalb eines Building Blocks kann einen Dienst anbieten, dessen Merkmale in einem Dienstvertrag festgelegt sind. Objekte aus verschiedenen Building Blocks interagieren nur über einen Dienstvertrag. Die Infrastruktur für diese Interaktion wird durch eine verteilte Prozeßumgebung zur Verfügung gestellt. Ein Building Block kann mehrere Dienstverträge anbieten. Er hat folgende Merkmale:

- *Einheit des Betriebs*: Ein Building Block ist als eine Einheit installiert und verwaltet.

- *Einheit der Verteilung*: Ein Building Block ist immer an einem Knoten des Netzwerks lokalisiert. Die Adressierung kann sich ändern. Um lokale Transparenz zu gewährleisten, ist jeder Dienstvertrag durch einen logischen Namen definiert, der unabhängig vom Ort des Building Blocks ist.

- *Einheit der Sicherheit*: Interaktionen über Building Blocks werden durch Sicherheitsüberprüfungen abgesichert, Interaktionen innerhalb von Building Blocks nicht.

- *Einheit der Interaktion*: Interaktionen über Building Blocks sind nur durch Dienstverträge möglich, die durch standardisierte Modelle, Protokolle und Notationen spezifiziert sind. Interaktionen innerhalb von Building Blocks können nichtstandardisierte Modelle und Protokolle benutzen.

2.2.3 Anordnung von Objekten in Building Blocks Eine grundlegende Methode, an die man sich bei der Anordnung von Objekten in Building Blocks halten sollte, ist das Segmentierungsprinzip. Dieses Prinzip besagt, daß Funktionalitäten, die von der Technologie zur Vermittlungs- und Übertragungsausrüstung benutzt werden, von den anderen Funktionalitäten getrennt werden sollen.

Die erste Kategorie nennt man Übertragungssegment Funktionen. Sie unterstützen die Vermittlung und die Übertragung von Information (z.B. Vermittlungsausrüstungskontrollfunktionen).

Die zweite Kategorie sind die Dienstsegment Funktionen. Sie befassen sich mit dem Management von Netzwerkressourcen und Endbenutzerdiensten (z.B. Abrechnungsverwaltung).

Dienstsegment Building Blocks und Übertragungssegment Building Blocks interagieren nur über Verträge. Diese Interaktion basiert auf dem Agentenmodell, das von Netzwerkmanagementstandards festgelegt wird. Jeder Übertragungssegment Building Block unterstützt Managementoperationen durch Sammlung von Information über Netzwerkressourcen, die als Managementobjekte modelliert sind. Diese Operationen sind in Verträgen angeordnet. Dienstsegment Building Blocks erfüllen ihre Managementfunktionen durch das Anrufen dieser Verträge. Übertragungssegment Building Blocks nutzen bei der Implementierung ihrer Verträge produktspezifische Technologien.

Diese Trennung erlaubt die Wiederverwendbarkeit von Dienstsegment Building Blocks über die verschiedenen Herstellerausrüstungen, die auf unterschiedlichen Technologien beruhen. Desweiteren erlaubt diese Trennung eine schnellere Einführung und Modifikation von Diensten.

2.3 Verteilte Prozeßumgebung

Eine Verteilte Prozeßumgebung (DPE, Distributed Processing Environment) unterstützt die erforderliche Infrastruktur, um Objekte aus verschiedenen Building Blocks interagieren zu lassen. Sie unterstützt verteilte Transparenz, indem die Komplexität der Verteilung vor den Objekten verborgen wird. Die meisten Komponenten der DPE sind mit Hilfe von Building Blocks und Vertragsprinzipien aufgebaut. So ist gesichert, daß die

einzelnen Komponenten nur über Standardprotokolle interagieren. Der Zweck der DPE ist das Anbieten von Diensten, die verteiltes Bearbeiten ermöglichen (vgl. Abbildung 80).

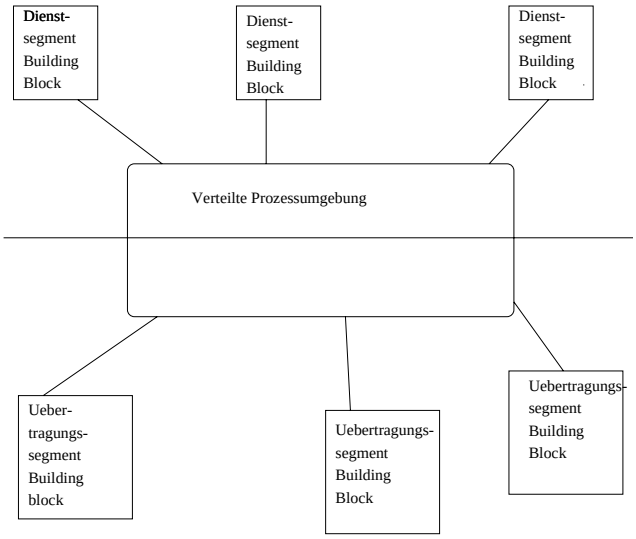

Abbildung 80. Dienstsegmentierung

Ein intelligentes Netzwerk kann in vier Ebenen untergliedert werden.

- Die unterste Ebene ist die grundlegende Rechen- und Kommunikationsumgebung. Diese besteht aus dem Betriebssytem und damit verbundenen Diensten, wie z.B. Diensten der Transportschicht, die Ende-zu-Ende Kommunikation unterstützen.

- Die nächste Ebene besteht aus der bereits vorgestellten DPE. Die DPE-Laufzeitumgebung besteht aus zwei Komponenten, dem DPE-Kern und den DPE-Dienstgebern. Der DPE-Kern unterstützt dabei Mechanismen zum Senden und Empfangen von Nachrichten. Bei DPE-Dienstgebern hingegen können die folgenden unterschieden werden:

  - *Vertragshändler*: Er unterstützt die Vertragsregistration und Vertragsauswahldienste.

  - *Authentisierungs-Server*: Dieser Server unterstützt die Dienste zur Authentisierung der Benutzer und der Building Blocks.

  - *Genehmigungs-Server*: Er unterstützt die Dienste zum Setup und zur Modifikation von Zugangskontrollen in Dienstverträgen.

  DPE Dienstgeber werden mit Hilfe von Building Blocks und Vertragsprinzipien aufgebaut. Jeder Dienstgeber unterstützt einen oder mehrere Verträge, die von Building Blocks oder anderen Dienstgebern genutzt werden können. Der DPE-Kern ist in jedem Netzwerkknoten vorhanden.

- Die nächste Ebene wird als Netzwerkressourcenmanagement bezeichnet. Sie befaßt sich mit dem Management und der Kontrolle von Netzwerkressourcen. Man unterscheidet zwei Funktionen:
  - *Systemmanagement-Funktionen*: Diese Funktionen unterstützen das Management von individuellen Netzwerkressourcen.
  - *Netzwerkmanagement-Funktionen*: Diese Funktionen benutzen Systemmanagement-Funktionen, um Strategien des Netzwerkressourcenmanagements durchzusetzen. Außerdem wird das Zusammenspiel verschiedener Netzwerkressourcen verwaltet.
- Die oberste Ebene ist die Dienstebene des Informationsnetzwerkes. Sie ist aus Dienstsegment Building Blocks zusammengesetzt. Diese unterstützen dienstspezifische Funktionen für Endbenutzer.

2.4 Exkurs: Die Architektur eines Telekommunikations Management Netzwerkes

Funktional kann ein Telekommunikations Management Netzwerk (TMN) in die folgenden fünf Schichten untergliedert werden [AKS93]:

- *Netzwerkelement-Schicht*: Hier werden alle Elemente modelliert, die das zu managende Netzwerk bilden.
- *Netzwerkelementmanagement-Schicht*: Sie ist verantwortlich für das Management einer Teilmenge der Netzwerkelemente im gesamten Netz.
- *Netzwerkmanagement-Schicht*: Diese Schicht ist für die technische Unterstützung der von höheren Schichten angeforderten Dienste verantwortlich.
- *Dienstmanagement-Schicht*: Sie ist für alle Verhandlungsergebnisse zwischen einem Benutzer und dem ihm angebotenen Dienst verantwortlich.
- *Unternehmensweite Management-Schicht*: Diese Schicht ist für das Management des gesamten Unternehmens verantwortlich.

3 Die Telekommunikations-Informations-Netzwerk Architektur (TINA)

3.1 Einführung

3.1.1 Das TINA-Konsortium Das TINA-Konsortium ist ein multinationales, weltweites Konsortium mit dem Ziel, eine offene Architektur für Telekommunikationsdienste zu definieren und diese zu validieren [DNI95]. Diese Architektur soll auf verteiltem Rechnen basieren und ist mit objektorientierten Konzepten sowie anderen Standards der Telekommunikations- und Computerindustrie in Verbindung zu bringen (z.B. IN, TNM). Seit der Gründung im Jahre 1992 in Japan arbeiten Ingenieure von mehr als 30 Unternehmen an diesem Projekt. Im Moment arbeiten mehr als 40 Ingenieure im zentralen Team in New Jersey. Die ersten beiden Teile der TINA-Architektur Spezifikation wurden in den Jahren 1993 und 1994 fertiggestellt, der dritte sollte Ende 1995 folgen.

3.1.2 Die Ziele des Konsortiums Viele Fragen, denen die Telekommunikationsindustrie gegenübersteht, sind weltweit ähnlich. Netzwerkbenutzer erwarten, daß Hindernisse abgebaut und sie an den Netzwerken beteiligt werden. Vielfach ist nicht mehr die Frage, wie man einen standardisierten Dienst unterstützt, sondern welchen und wann. Das Intelligente Netzwerk (IN) und das Telekommunikations Management Netzwerk (TMN) sind bereits zwei weitgehend anerkannte standardisierte Versuche, Teile dieser Fragen zu lösen. Sie müssen jedoch zusammen betrachtet werden.

Die Vision des TINA-Konsortiums ist eine einheitliche Architektur für die offene Telekommunikation. Diese Architektur soll sich nach dem Bedarf der traditionellen und der neuen Dienste sowie dem der Management- und Operationsdienste richten und flexibel für neue Technologien sein. Die Softwarearchitektur sollte dazu folgendes anbieten:

- Wiederverwendbarkeit von Softwarekomponenten
- Unterstützung netzwerkweiter Softwarebenutzung
- leichte Konstruktion, leichter Test und leichter Einsatz der Dienste
- Verbergen der Komplexität der zugrundeliegenden Technologie und der durch die Verteilung hervorgebrachten Komplexität vor dem Programmierer

Die neuesten Fortschritte im verteilten Rechnen und der objekt-orientierten Analyse sollen zu diesem Zweck verwendet werden.

3.2 Die TINA-Architektur

Die TINA-Architektur ist in die drei Bereiche Computing-, Dienst- und Managementarchitektur unterteilt. Jeder dieser Bereiche wird in einem der folgenden Abschnitte vorgestellt.

Die generelle Architektur der Dienste ist dabei so, wie sie in Kapitel 2 bereits beschrieben wurde. Um eine gute Strukturierung zu erreichen, werden die regulären (nicht DPE)

Dienstkomponenten in drei Kategorien unterteilt, die aktuellen Dienstkomponenten, die in der Dienstarchitektur beschrieben werden, die Ressourcenmanagement Komponenten und die Ressourcen Komponenten, die in der Managementarchitektur beschrieben werden (vgl. Abbildung 81).

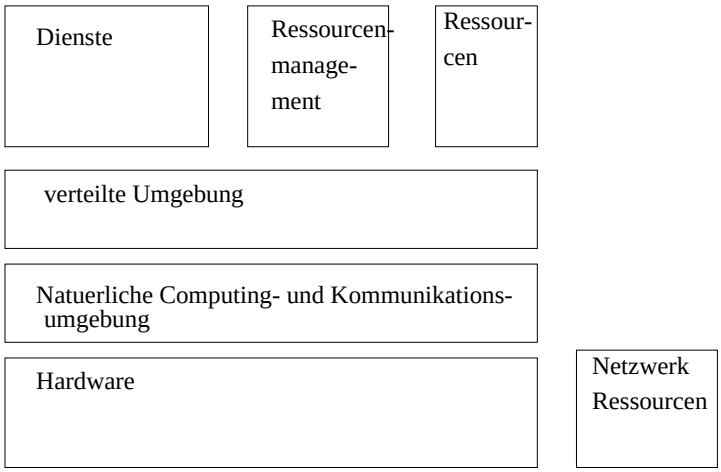

Abbildung 81. generelle Architektur

3.2.1 Die Computingarchitektur Eines der ersten Prinzipien der TINA-Computingarchitektur ist die Nutzung der Objektorientiertheit. Als eine Verbesserung des RM-ODP Objektmodells wurde ein zusammengesetztes Modell mit einer sinnvollen Trennung der Bereiche einer Softwarekomponente unterstützt. Dieses Modell (USCM) besagt, daß jede Softwarekomponente aus einem Kern, der das Modell grundsätzlich beschreibt, einem Nutzungsteil, der das Erscheinungsbild für den Benutzer beschreibt, einem Managementteil, der Operationen, Wartung und Verwaltung der Komponente unterstützt, und einem Substanzteil, der die Abhängigkeit zu anderen Komponenten beschreibt, besteht.

Ein anderes Prinzip der TINA-Computingarchitektur besagt, daß Telekommunikationsdienste, Telekommunikationsmanagementdienste und verteilungsspezifische Gesichtspunkte als softwarebasierte Anwendungen betrachtet werden.

Das Interaktionsmodell schreibt dabei die Regeln vor, nach denen eine Softwarekomponente mit einer anderen interagieren kann (computationel modeling concept). Die Softwarekomponenten interagieren über Schnittstellen, wobei jede Komponente Dienste an mehreren Schnittstellen anbietet und Dienste anderer Schnittstellen aufrufen kann.

Bei den Schnittstellen kann zwischen verarbeitenden Schnittstellen und Datenflußschnittstellen unterschieden werden. An verarbeitenden Schnittstellen können Interaktionen in Bezug auf den Aufruf der Operationen und die Antwort darauf gegliedert werden.

Datenflußschnittstellen hingegen repräsentieren Kommunikationsendpunkte oder verarbeiten einen Informationsfluß. Die Schnittstellen sind unabhängig von einer Programmiersprache durch eine Spezifierungsnotation festgelegt. Wie bereits in Kapitel 2 beschrieben, liefert die verteilte Prozeßumgebung (DPE) dabei die Grundlagen für die softwarebasierten Anwendungen.

3.2.2 Dienstarchitektur Die grundlegende Computingarchitektur wurde erweitert um Konzepte und Prinzipien, die in der Dienstarchitektur zusammengefaßt sind. Diese Dienstarchitektur stellt Mittel zur Verfügung, um Dienste und eine Dienstunterstützungsumgebung zu bilden. Sie ist anwendbar auf eine Menge von Diensttypen, z.B. auf Managementdienste, Informationsdienste, Transportdienste und Zugangsdienste.

Die Dienstarchitektur enthält eine Definition der Grundsätze, über die nachgedacht werden muß, wenn ein Dienst definiert wird, und eine Definition über die Rolle, die diese Dienste spielen. Die wichtigsten Rollen sind die des Benutzers, des Teilnehmers, des Netzwerkbetreibers, des Dienstanbieters, des Dienst/Netzwerk-Designers und des Dienst/Netzwerk-Managers. Es wurde auch ein Lebenszyklusmodell entworfen, in dem fünf Zustände eines Dienstes unterschieden werden:

- der Bedarf eines Dienstes
- die Konstruktion eines Dienstes
- die Einrichtung eines Dienstes
- die Ausführung eines Dienstes
- die Zurücknahme eines Dienstes

Die Dienstarchitektur identifiziert außerdem die wichtigsten Laufzeitkomponenten der meisten Dienste. Dies erlaubt es, eine Bibliothek von wiederverwendbaren Komponenten aufzubauen, die nützlich sind, um weitere Dienste zu entwickeln. Diese Komponenten bestehen unter anderem aus:

- generischen Endpunkten für Sitzungen
- Benutzeragenten
- Managern für Dienstsitzungen
- Managern für Kommunikationssitzungen

Ein Informationsmodell und ein Ausführungsmodell sind bereits verfügbar. Die Dienstarchitektur tritt ferner für eine Trennung zwischen Dienstlogik und Verbindungsverwaltung auf der einen sowie Ressourcenbereitstellung, -kontrolle und -management auf der anderen Seite ein. Diese Trennung erlaubt es, Dienste unabhängig vom Netzwerk zu entwerfen, und aus dem Netzwerkbetrieb erhöhten Gewinn zu erzielen, indem mehr Dienste angeboten werden können.

3.2.3 Managementarchitektur Die Managementarchitektur spezifiziert eine Menge von wiederverwendbaren Managementfunktionen in Bezug auf Netzwerkmanagementdienste, Dienstmanagementdienste und DPE-Managementdienste.

Desweiteren wurden wichtige Richtlinien und Prinzipien in diese Architektur aufgenommen, wie beispielsweise die Schichtenstruktur des TMN-Modells, die OSI-basierte funktionale Trennung der Managementbereiche sowie das Paradigma der Managed Objects.

Aus der Sicht der Informationsbehandlung führt die Anwendung der oben genannten Prinzipien zur Betrachtung von Managementdiensten und ihre gemanageten Ressourcen als m:n-Relationen. Eine Managementressource, wie z.B. ein Transportnetzwerk, wird modelliert als eine Zusammensetzung von Teilnetzwerken, Verbindungen, Endpunkten und Wegen, die Endpunkte durch Aneinanderreihen von Teilstrecken verbinden.

Aus der Sicht der Ausführung führt die Anwendung der genannten Prinzipien dazu, daß so viele Softwarekomponenten definiert werden, wie funktionale Managementbereiche (OSI) pro TMN-Schicht vorhanden sind. So tritt die TINA-Managementarchitektur z.B. für eine Trennung zwischen dem Verbindungsverwalter auf der Netzwerkmanagement-Schicht und dem auf der Netzwerkelementmanagement-Schicht ein. Die zur Zeit spezifizierten Softwarekomponenten umfassen Verbindungs-Managementdienste, Ressourcen-Konfigurationsdienste, Fehler-Managementdienste, CSPS DPE-Managementdienste und Beschreibungsdienste.

Das nächste Kapitel beschreibt als ein Ergebnis von TINA das Netzwerk Ressourcen Modell.

3.3 Das TINA Netzwerk Ressourcen Modell

3.3.1 Zweck Das Netzwerk Ressource Informations Modell (NRIM) definiert ein Modell der Netzwerkressourcen, das der Dienstsoftware die notwendige Abstraktion zur Verfügung stellt, um die Ressourcen des Netzwerks zu benutzen und zu verwalten [LPW+96]. Dienstsoftware nutzt das NRIM, wenn Netzwerkverbindungen aufgebaut werden müssen, und wenn sie für das Management von Netzwerkressourcen verantwortlich ist. Das Modell unterstützt eine vollständige Sicht der Verbindungen auf allen Netzwerkebenen. Die Sicht der obersten Ebene wird für Anwendungen genutzt.

Die Ziele der NRIM-Spezifikation sind, die Managementressourcen zu spezifizieren, die den Dienstanwendern zur Verfügung gestellt werden, die Managementressourcen zu spezifizieren, die den Netzwerkmanagementanwendungen zur Verfügung gestellt werden, und die Wiederverwendbarkeit der Dienste und der Netzwerkmanagement-Anwendungssoftware zu ermöglichen. Das TINA Netzwerk Ressourcen Modell bietet eine generische Betrachtungsweise von Netzwerkressourcen wie Teilnetzwerken, Netzwerkelementen, Wegen, Verbindungen und Endpunkten aus Sicht der Informationsbehandlung. Die grundsätzlichen Ziele des NRIM sind dabei:

- verschiedene Diensttypen zu unterstützen
- von der Übertragungs- und Vermittlungstechnologie unabhängig zu sein
- bestehende Standards bestmöglich zu nutzen

3.3.2 Nutzung von bisherigen Ergebnissen ITU-T Empfehlung G.803 besagt, daß jedes Netzwerk aus einer Anzahl von geschichteten Netzwerken besteht. Diese geschichteten Netzwerke können in Subnetzwerke und Verbindungen zerlegt werden. Subnetzwerke bestehen wiederum aus kleineren Subnetzwerken und zugehörigen Verbindungen. Am Ende der Zerlegung erhält man ein Netzwerkelement. Die Verbindungsbeziehung der

Dienstgeber/nehmer Beziehung zwischen benachbarten Schichten wird durch einen Weg des Netzwerks der Dienstschicht zur Verfügung gestellt.

Standard M.3100 zum generischen Netzwerkinformationsmodell erklärt eine Reihe von generischen Objektüberklassen, wie das Managed Element und den Endpunkt.

Die größte Ergänzung der TINA zu den oben genannten Modellen ist das Konzept des Verbindungsdiagramms. Es zeigt ein Modell der Verbindungsmöglichkeiten zwischen Endpunkten, unabhängig davon, wie es geschaffen wurde, und unabhängig von der zugrundeliegenden Netzwerktechnologie.

Das TINA NRIM definiert auch andere Objekttypen wie:

- die Verbindungsgruppe: Diese hat zum Ziel, Gruppen von Verbindungen zu managen, die synchronisiert werden müssen.
- Kante: Sie geht mit Mobilität und mehrartigen Verbindungen um.

3.3.3 Das Modell Das NRIM-Modell besteht aus einer Reihe von Bereichen. Bisher wurden neun solcher Bereiche definiert. Als ein Beispiel wird der Verbindungs-Modellbereich beschrieben.

Der Verbindungs-Modellbereich enthält vor allem Objekte der Ressourcenebene. In Bezug auf Verbindungsmöglichkeiten können innerhalb des geschichteten Netzwerks zwei verschiedene Verbindungstypen unterschieden werden:

- Subnetzwerkverbindung (SNC): Sie beschreibt die Verbindungsmöglichkeit in einem Subnetzwerk und ist flexibel beim Auf- und Abbau als Teil des Verbindungsmanagements.
- Verbindung: Sie beschreibt die Verbindungsmöglichkeit zwischen zwei Subnetzwerken und ist unflexibel in Bezug auf das Verbindungsmanagement.

Die Topologische Verbindung und ihre Einzelverbindungen werden vom Ressourcen-Konfigurationsmanagement verwaltet, Subnetzverbindungen hingegen durch das Verbindungsmanagement.

Die ITU Empfehlung G.803 führt ein Tandem-Verbindungsobjekt ein, das seriell verbundene Verbindungen und Subnetzverbindungen identifiziert. Während ein Weg(objekt) nur Wissen über seine Endpunkte besitzt, hat eine Tandem -Verbindung das Wissen über alle seine individuellen Subnetzverbindungen und deren Verbindungen, die zusammen die Ende-zu-Ende Verbindungsmöglichkeit bieten.

Ein Kanten-Objekt bindet einen existierenden Endpunkt an eine Subnetzverbindung. Die Endpunkte werden dabei durch die Ressourcenverwaltung zur Verfügung gestellt und verwaltet. Instanzen der Kanten-Klasse werden vom Verbindungsmanagement beim Aufbau einer Verbindungen gebildet und gelöscht, wenn die Verbindung abgebaut wird.

Bei Einrichtung einer Subnetzverbindung wird gleichzeitig ein Kanten-Objekt erzeugt und an den quellseitigen Endpunkt gebunden. Ein Verbindungsobjekt hat somit folgendes Aussehen.

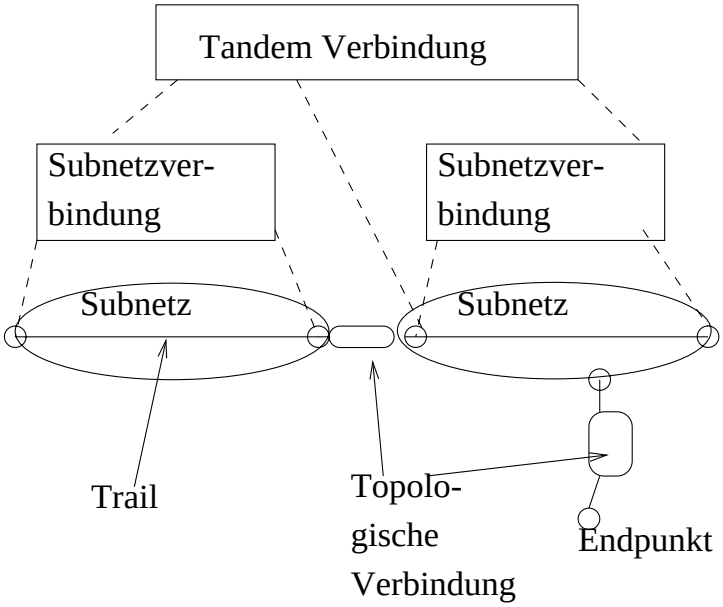

Abbildung 82. Verbindungsobjekt

4 Zusammenfassung

Dem TINA-Konsortium ist es gelungen, eine Softwarearchitektur zu definieren, die bereits auf den wichtigen Standards des verteilten Arbeitens aufbaut.

Sie besteht aus einer Computing Architektur, die die Basis für alle verteilte Telekommunikationssoftware-basierten Anwendungen bietet, einer Dienstarchitektur, die Konzepte der Dienstbildung, -entwicklung und -bearbeitung hinzufügt, und einer Managementarchitektur, die wiederverwendbare Managementdienste erkennt und Richtlinien zur Entwicklung von Managementdiensten und Managementressourcen definiert hat. Die TINA-Architektur will dem Telekommunikationsanbieter helfen, Breitband-Multimediadienste effizient zu entwickeln.

Neben einer allgemeinen Vorstellung der TINA-Architektur präsentiert dieser Beitrag das vom TINA-Konsortium entwickelte Netzwerk Ressourcen Modell (NRIM). Das NRIM baut auf bestehenden Konzepten auf und unterstützt eine Reihe neuer Multimedia Dienste, indem es ein Modell der Netzwerkressourcen definiert, das der Dienstsoftware die notwendige Abstraktion zur Verfügung stellt, um die Ressourcen des Netzwerks zu benutzen und zu verwalten.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[10192] ISO/IEC CD 10164-13. *Information Technology - Open Systems Interconnection - Systems Management - Part 13: Summarization Function.* Committee Draft. 1992.

[AC95] Peter Alexander und Kacey Carpenter. ATM Net Management: A Status Report. *Data Communications*, Sep 1995, Seite 110–116.

[AF93] The ATM-Forum. BISDN Inter Carrier Interface (B-ICI) Specification, Version 1.0. August 1993.

[AF95] The ATM-Forum. Interim Local Management Interface (ILMI) Specification Version 4.0. *ATM-Forum/095-0417R2*, Okt 1995.

[AKS93] Menso Appeldorn, Roberto Kung und Roberto Saracco. TMN + IN = TINA. *IEEE Communications Magazine*, März 1993, Seite 8.

[ANS94] ANSI. B-ISDN Operations and Maintenance Principles and Functions (T1s1.5/93-004R2). Januar 1994.

[Arm96] G. J. Armitage. Support for Multicast over UNI 3.0/3.1 based ATM Networks. Bellcore, February 1996. IETF Internet-Draft.

[AT94] M. Ahmed und K. Tesink. Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2. *RFC 1695*, Aug 1994.

[Bel94] Bellcore, SR-NWT-002268. *Cycle 1 Initial Specifications for INA*, Juni 1994.

[BHK95] A. Badach, E. Hoffmann und O. Knauer. *High Speed Internetworking.* Addison-Wesley. 1995.

[Bir87] Joseph Birman. Reliable communication in the present of failures. *ACM Trans. Comput.* **5**(1), Feb. 1987, Seite 47–76.

[Bre91] Hans-Jürgen Breuer. ATM-Layer OAM: Principles and Open Issues. *IEEE Communications Magazine*, Sep 1991, Seite 75–81.

[CA86] Christian und Aghili. Strong Atomic Broadcast. *IBM Research Report* **RJ 5244**(54244), Juli 1986.

[Cci89] *CCITT Recommendation M.30, "Principles for a telecommunications management network".* Blue Book, vol. IV, Fascicle IV.1, 1989.

[Cci92] *CCITT Draft Recommendations M.3xxx Series*, 1992.

[CM84] Chang und Maxemchuck. Reliable broadcast protocols. *ACM Trans. Comput.* **2**(3), Aug. 1984, Seite 251–273.

[Dee89] S. Deering. Host Extensions for IP Multicasting. Stanford University, August 1989. RFC 1112.

[dIF95] Autorenkollektiv des ITG-Fachausschusses 1.7. Netzmanagement. *ntz* Band 6, Jun 1995, Seite 6–48.

[DNI95] Fabrice Dupuy, Gunnar Nilsson und Yuji Inoue. The TINA Consortium. *IEEE Communications Magazine*, November 1995, Seite 6.

[GMP96] W. Greene, M. Maston und A. Prasad. draft-ietf-atommib-acct-02.txt. *INTERNET DRAFT*, Jun 1996.

[GP96] W. Greene und A. Prasad. draft-ietf-atommib-acct-01.txt. *INTERNET DRAFT*, Apr 1996.

[JBI93] William J.Barr, Trevor Boyed und Yuji Inoue. The TINA Initiative. *IEEE Communications Magazine*, März 1993, Seite 7.

[Kau95] Franz-Joachim Kauffels. *Netzwerk- und System-Management: Probleme - Standards - Strategien.* DATACOM Buchverlag GmbH. 1995.

[Kya95] Othmar Kyas. *ATM-Netzwerke: Aufbau - Funktion - Performance.* DATACOM-Verlag. 1995.

[Lau93] M. Laubach. Classical IP and ARP over ATM. Hewlett-Packard Laboratories, December 1993. RFC 1577.

[LPW+96] Magnus Lengdell, Juan Pavon, Masaki Wakano, Martin Chapman und Motoharu Kawanishi. The TINA Network Resource Model. *IEEE Communications Magazine*, März 1996, Seite 6.

[Lud87] T. Ludwig. Parallelisierung des Monte Carlo Verfahrens nach Bird. *Universität Erlangen-Nürnberg* Band Teilprojekt B1, 1987.

[Mey90] Bertrand Meyer. *Objektorientierte Softwareentwicklung.* Carl Hanser Verlag München Wien. 1990.

[MK94] Keith McCloghrie und Frank Kastenholz. Evolution of the Interfaces Group of MIB-II. *RFC 1573*, Januar 1994.

[MM89] F. Mattern und H. Mehl. Diskrete Simulation - Prinzipien und Probleme der Effizienzsteigerung durch Parallelisierung. *Informatik-Spektrum* (12), 1989, Seite 198 – 210.

[NT96] Michael Noto und Kaj Tesink. Definitions of Tests for ATM Management. *Internet Draft <draft-ietf-atommib-test-00.txt>*, Januar 1996.

[Pra96] Anil Prasad. draft-ietf-atommib-acct-00.txt. *INTERNET DRAFT*, Feb 1996.

[Pyl94] Raymond H. Pyle. Applying Object-Oriented Analysis and Design to the Integration of Network Management Systems. In *Telecommunications network management into the 21 century: techniques, standards, technologies, and applications.* Institute of Electrical and Electronics Engineers, Inc., 1994.

[SA96] T. J. Smith und G. Armitage. IP Broadcast over ATM Networks. IBM Corporation, Bellcore, April 1996. IETF Internet-Draft.

[Sei94] Jochen Seitz. *Netzwerkmanagement.* Thomson's Aktuelle Tutorien, International Thomson Publishing. 1994.

[Sie94] Gerd Siegmund. *ATM - Die Technik des Breitband-ISDN.* v. Decker. 1994.

[SMSP96] Schill, Mittasch, Spaniol und Popien (Hrsg.). *"Distributed Platforms" Proceedings of the IFIP/IEEE International Conference on Distributed Platforms: Client Server and beyond: DCE, CORBA, ODP and Advanced Distributed Applications.* Chapman & Hall, 1996. ISBN: 0-412-73280-7.

[Sta94] William Stallings. Kerberos Keeps the Enterprise Secure. *Data Communications*, Oct 1994, Seite 103–111.

[Sta95] William Stallings. *Network and Internetwork Security.* Prentice Hall. ISBN: 0-7803-1107-8, 1995.

[STK94] A. Schade, P. Trommler und M. Kaiserswerth. Object Instrumentation for Distributed Applications Management. In *Telecommunications network management into the 21 century: techniques, standards, technologies, and applications.* Institute of Electrical and Electronics Engineers, Inc., 1994.

[Tan96] Andrew Tannenbaum. *Computer Networks.* Prentice Hall. 1996.

[Yan93] Elliot Limin Yan. The Design and the Implementation of an Emulated WAN. 1993.

[Zit95] Martina Zitterbart. *Hochleistungskommunikation.* R. Oldenburg Verlag München. 1995.